The Pragmatic Programmers

TURING 图灵程序设计丛书

Web Design for Developers

A Programmer's Guide to Design Tools and Techniques

写给程序员的Web设计书

[美] Brian P. Hogan 著

吴珂 译

- 会开发，也要懂设计
- 领悟先进的Web设计理念
- 让你的网站熠熠生辉



人民邮电出版社

POSTS & TELECOM PRESS

Web设计常常面对着一个很大的挑战，它需要结合设计人员与程序员的思维，要跨越纯视觉思考者和纯线性思考者之间的交流鸿沟。现实世界中，很多网站就是程序员设计的作品，如何让程序员拥有设计者的视角，了解配色、字体、页面布局等知识，并熟悉相关技术和设计工具，是本书重点讲述的内容。

本书字里行间融汇了先进的Web设计思想，让开发人员得以避免网页设计误区，真正提高设计能力，开发出爽心悦目的网站。

作者简介

Brian P. Hogan 是一名自由职业者和开发顾问。他从1995年就开始从事专业网站开发，使用ASP、PHP和Ruby开发过各种类型的网站和Web应用程序。他喜欢传授技术知识，撰写技术文章，尤其擅长网页设计和开发方面的技术。

图书在版编目（C I P）数据

写给程序员的Web设计书 / (美) 霍根 (Hogan,B.P.)
著 ; 吴珂译. -- 北京 : 人民邮电出版社, 2011.8
（图灵程序设计丛书）
书名原文: Web Design for Developers : A
Programmer’s Guide to Design Tools and Techniques
ISBN 978-7-115-25911-0

Ⅰ. ①写… Ⅱ. ①霍… ②吴… Ⅲ. ①网页制作工具
－程序设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2011)第129425号

内 容 提 要

本书系统而深入地阐释了网站的设计与实现，帮助读者从开发人员的角度理解什么是设计。通读本书之后，读者可以跨越纯视觉思考者和纯线性思考者之间的交流鸿沟。本书的主要内容有：如何挑选配色，如何选择字体，如何用 Photoshop 实现基本设计，如何创作 Banner 等页面元素，如何制作 HTML 和 CSS 模板以及如何测试设计的兼容性和可访问性等。

本书适合开发人员和 Web 设计师研读，对于那些独立且没有设计背景的开发人员非常有用。

图灵程序设计丛书
写给程序员的Web设计书

◆ 著 [美] Brian P.Hogan
译 吴 珂
责任编辑 明永玲
执行编辑 丁晓昀

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京 印刷

◆ 开本：800×1000 1/16
印张：16.25 彩插：8
字数：341 千字 2011年 8 月第 1 版
印数：1 – 3 000 册 2011年 8 月北京第 1 次印刷

著作权合同登记号 图字：01-2009-5710号

ISBN 978-7-115-25911-0

定价：49.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

读 者 评 论

真希望我在第一次做网站的时候就能看到这本书。本书涵盖了 Web 开发的方方面面，当你需要改进网站时，这本书能为你解答很多问题。

——Shae Murphy，Social Brokerage CTO

作为 Web 开发人员，我自以为了解 HTML 和 CSS。这本书让我认识到，只了解那些基础知识是不够的，Web 设计决不只是改改字体和颜色那么简单。

——Mike Weber，Web 应用开发人员

如果你已经准备好要踏入 Web 设计的奇妙世界，读这本书可以让你清晰且有效地了解那些关键概念。另外，轻松的行文风格也使得阅读本书是一种享受。

——Jeff Cohen，Purple Workshops 创始人

无论是初学者还是经验丰富的设计师都能在这本书中有所收获。从开发人员的角度看，这本书在我的日常工作中发挥了巨大的作用，它让我在组织页面内容的时候三思而后行。

——Chris Johnson，解决方案开发专家

从概念入手，Hogan 引领读者见识了网页设计的各个阶段，内容完备而专业。初学者和行业老手都能从本书中获益，收获技术以及技术之外的各种知识。

——Neal Rudnik，Aspect，Web 和多媒体产品经理

在这本以开发人员为目标读者的书中，模糊了一些公司中存在的设计团队和开发团队之间的界限。毕竟，“程序员”也可以创造出美观易用的页面。

——Jon Kinney，Avastone Technologies，Ruby 构架师

本书重点介绍了简单实用的技巧，就算你对对称的理解仅仅局限于程序里的大括号，运用书中所学的技巧也能让你的网页看起来很专业。

——Craig Castelaz，一家大公司里的小软件工程师

第四部分　准备上线

第 1 章 引 言

如果你曾经写过 Web 应用程序并想让它变得更漂亮，如果你想知道你中意的网站是如何用 CSS 拼凑起来的，那么这本书就是为你准备的。但是，如果你想问的是，舔多少下才能把夹心棒棒糖的巧克力糖心舔出来，本书就无能为力了，维基百科才是你应该去的地方。

这本书是为没有设计背景的程序员准备的，它涵盖了 Web 设计的主要过程。在那些光鲜亮丽、布局合理的网站背后，凝结的是网页设计师们辛勤的汗水。设计网站就像用 Java、Ruby 或者 C#写程序一样，只有遵守一定的规则和良好的习惯，你才能获得想要的结果。

1.1 说在前面的话

好的 Web 设计不仅仅指漂亮的页面，像色彩、字体、布局和可用性等一些概念也是设计的一部分。只有当这些元素都被考虑到了，设计出来的网站才会让用户眼前一亮。也许你挑对了颜色和渐变的搭配，却没有顾及字体的可读性，那么设计出来的仍然是一个糟糕的网站；也许你能在 Photoshop 或者 GIMP①中做出一个好看的图像，但对 HTML 和 CSS 不够了解，你也无法在浏览器中实现你的设计。如果你的标记语言结构混乱，JavaScript 就会漏洞百出；如果你不优化页面内容，搜索引擎可能就会忽视它；如果你完全没有考虑可访问性和可用性，用户甚至会厌恶你的网站。

条条大路通罗马

本书是为那些想要了解 Web 设计知识的程序员写的,涵盖了一些相当有用的入门级 Web 设计知识。当然，书中所提到的方法并不是设计网页的唯一路径，但是其中讲到的技术会让你具备足够的知识去探索网页设计中的新领域，并最终形成自己的风格。

① Linux 中为 X Windows 准备的图像图形软件。——译者注

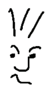
> 你对书里的许多例子可能有不同的见解，会想到用不同的方法或者技术去解决那些问题，这样很好。我故意留下这些“路口”，因为这是成为设计师的必由之路。慢慢地你就会发现自己的变化，同时了解用户的需求变化。我很期待看到你的设计作品。

另外，创意同样是成功的 Web 设计中不可或缺的一环。因此我希望在做本书中的练习时，你能尽量发挥自己无穷的想象力。在书中，我会展示完整的 Web 设计流程，希望你不会受我所说的东西限制，能在本书的指导下形成自己的风格，比如选择自己的字体和颜色。这样，你才能从理论中学到实用的东西。我希望你设计出来的网页可以跟我举的例子完全不同。

编程经验有助于你设计出吸引眼球的页面。书的前半部分是一个设计的世界，你会学到颜色、字体等方面的知识，这些概念是网页设计的基础；另外，这部分还会介绍设计师们常用的工具和技术。在有了足够的理论储备之后，如何写代码将成为本书的主要内容。毕竟，这是一本给程序员看的设计书，对吧？

1.2 网页设计实战

让我们跟着 Web 开发人员 Ron 来学习网页设计的主要流程吧。Ron 很忙，他刚刚从一个客户那里接了一个设计小网页的活儿。



小乔爱问……

现在还有人在 Photoshop 里做页面样式图吗

是的，设计师们依然是这么做的。如果你在办公室并没有看到这样的人，可能是因为他们都是写 CSS 的程序员，不是网页设计师。其实很大一群开发人员的日常工作就是从图形设计师那里接收 Photoshop 文件（PSD 文件），然后将这些 PSD 文件整合到 Web 应用中。知道如何处理 PSD 文件是网页设计过程中很重要的一环。

本书之所以用 Photoshop 来做页面样式示意图，原因有两个：一是 Photoshop 做出来的样式图很适合用来讲解设计过程中的不同环节，另一个原因是做完配好颜色的样式图之后，学习 CSS 的概念会容易不少。

1.2.1 明确要求

Ron 的新客户是一个房产中介，她要做一个小型的内容管理系统来管理房产信息。跟这个叫

Kim的中介碰过一次面之后，Ron就开始在纸上画首页的草图。根据客户的需求，Ron做了好几个设计，并从中选出了三个他认为最能满足 Kim 要求的方案。

然后 Ron 跟 Kim 又碰了一次面，Kim 从三个方案中选了一个，并提出了一些修改建议。讨论到配色的问题时，Kim 要求主题色调必须是蓝、灰和白，这样就能跟她名片的样式协调。

1.2.2 Photoshop时间

第二次会面之后，Ron 就开始在电脑前摆弄 Photoshop 了。他很快就按照 Kim 要求的方案和色彩做出了一个页面样式，从素材库里挑出了几张免版税的图片，然后尝试了几种不同的灰和蓝的组合，直到满意为止。最后从 Photoshop 里把图片导出，发给了 Kim。

一个星期之后，Ron 电话联系了 Kim，问她对目前为止的工作有何建议。Kim 说等她度假回来之后给他答复，让 Ron 再等一周左右。

1.2.3 代码时间

一周后，Ron 接到了 Kim 的电话，她对现在的样式很满意，并说可以进行下一步工作了。Ron 松了一口气，然后开始在自己喜欢的文本编辑器中写代码，把样式图变成网页。

Ron 从创建简单的 HTML 文档开始，确定页面的框架和内容，然后在 Photoshop 中把 Banner[①] 和其他一些图像切出来放到 HTML 文档中。

做完这些之后，Ron 写了一些 CSS 代码，将页面的整个内容组织起来。最终，他把一个竖版的页面框架变成了色彩明亮的双栏布局。

在 Firefox 浏览器中，他做的网页就跟样式图里的一模一样，很漂亮。但是，同样的网页在 IE6 中就面目全非了。

好在 Ron 曾经遇到过同样的情况，他娴熟地在专门针对 IE 的样式表中插入几行代码。终于，所有的事情都搞定了，Ron 把完成的网页发给了 Kim。

1.2.4 一切就绪

Kim 很喜欢那个首页，于是 Ron 开始做网站的其他页面。因为已经确定了颜色搭配、图像和样式表，剩下的工作就很简单了。Ron 自己也很开心，因为他的客户对他的工作很满意。

---

① 通常指网页顶端的横幅图像。——译者注

1.2.5 现实不一定总是如此美好

这一次 Ron 很走运，遇到了一个容易相处的客户，然而并不是所有的客户都这样。本书将以 YourFoodbox.com 为例进行讲解，Foodbox 网站的老板们就很难缠，你马上就能见识到了。

1.3 YourFoodbox.com

你要为一家资金非常充足的公司做一个大全式的菜谱分享网站，用户可以在这一网站上查找数以千计的各式菜谱，创建自己的菜谱以及修改已有的菜谱。你刚刚完成网页设计，在一周之内网站就要上线了，于是你将做好的东西给网站的老板们看。结果他们虽然对功能部分没有意见，却不能忍受网站的外观。他们可能会说"感觉差点什么"或"不能吸引眼球"之类的话。而且，他们一定不会提出具体的修改意见。你必须运用自己收集客户需求的经验，来尽量满足他们的需要。

本书里有几章会告诉你如何面对这种很常见的情况。你还会学到包括配色、字体、按钮设计、图像优化和网格模板设计等在内的一系列页面设计知识。你不但会了解如何让 Web 表单看起来更美观，还会接触到种类繁多的窍门，这些窍门能让页面完美地跨浏览器、跨平台。在完成页面搭建之后，搜索引擎优化和页面内容优化的技巧则能帮助你的页面在搜索结果里名列前茅，并且运行起来更顺畅。

同时，你还会了解网站可访问性的重要性，你需要让尽可能多的用户能更方便地使用你的网站，也要照顾到生理有缺陷人群的使用。这一点在商业上很重要，同时从我个人的角度来讲，我也很在意可访问性，因为我、我父亲和我女儿都是先天性白内障患者。当然，本书后半部分才会深入涉及可访问性的问题，但是在那之前的某些例子中，我可能会提到与可访问性及易用性相关的一些概念。

1.4 准备好了吗

虽然在我们面前的是一条很长的路，但是我们的目标明确。那么，就让我们从现有 YourFoodbox.com 页面的缺陷开始，看看我们的客户都有什么修改要求吧！

1.5 致谢

没有人能独自出版一本书。实际上，在出书的过程中，写作只占整个工作量的一小部分。这本书的出版离不开来自同事、朋友、家人的反馈、批评、友善和情感支持。

首先要感谢 Dave Thomas 和 Andy Hunt，谢谢他们签下了这本书。他们从一开始就对它充满信心，并且一直支持我。我从他们及 Pragmatic Bookshelf 出版的其他书里，学到了很多。和他们一起工作是我的荣幸。

然后要感谢 Daniel Steinberg，本书耐心又睿智的编辑。他一直都非常支持我，让我发现我写的东西其实还不赖，没我自己想的那么糟。这都多亏了他及时的反馈和适当的批评指正。

还要感谢本书的技术审稿人：Jeremy Sydik、Jon Kinney、Chris Johnson、Ben Kimball、Josh Peot、Mike Mangino、Lyle Johnson、James Wylder、Jeff Cohen 和 Mike Weber。谢谢他们肯花时间提出宝贵意见，好让我能在书中把自己的意思表达得更清楚。

另外要特别感谢 iStockphoto.com 的工作人员，感谢他们让我用他们图片库里的图片做本书中的示例。

感谢 Bruce Tate，他凭一己之力改变了我的职业生涯。

感谢威斯康星大学欧克莱尔（Wisconsin-Eau Claire）校区的 Lillian Hillis、Erich Tesky 和 Marian Ritland，感谢来自他们的支持、质疑和解答，愿我们之间的友谊长存。特别感谢 Marian 创造了一个可以让我们学习、成长和面对挑战的环境。

感谢 Bobby Pitts 的教导，让我真正学会了使用设计工具。每次拿起工具箱中的画笔工具，我都会情不自禁地想起他给我们上课的那些日子。

谢谢 Chris Warren、Kevin Gisi、Gary Crabtree、Carl Hoover、Josh Anderson 和 Adam Ludwig，让我有机会做他们的指导老师，帮助他们成长。他们的成功让我感到骄傲。

谢谢我的爸爸和 Claudia，谢谢他们的支持和建议。谢谢我的妈妈，是她把我带到这个世界上，很可惜她没能看到这本书的出版。

最后，感谢我的妻子 Carissa，还有我的女儿 Ana 和 Lisa，没有她们的爱和支持，我不可能写出这本书。能有这样一个完美的家庭是我此生的幸运。就算我因为要写作不能跟她们共度周末时光，她们还是一直鼓励我。谢谢你们，谢谢你们能如此支持我。我爱你们。

Part 1

第一部分

设计基础

本部分内容

第 2 章

网页（再）设计的基础——重新设计 Foodbox

本书的示例网站——Foodbox 是一个网络社区，用户可以在上面上传和分享菜谱。网站想做成市面上那些流行的社交网站的样子，用户到时可以给菜谱贴标签、留评论，并且能做出自己的烹饪书。

网站具有充足的资金支持，同时还拥有一群很有天赋的开发人员。你的同事 Steve 刚刚去给网站老板们过目了演示产品，回来后把画满圈圈点点的笔记本扔在了你的桌子上。

“他们很不喜欢首页的样子，”Steve 说，“从 Banner 到配色都不喜欢。他们说整个首页太素了，如果不按照这个单子上的意见修改，剩下的页面他们看都不想看了。”

2.1 目前的网站

先看看现在的页面是什么样的（参见图 2-1），然后再看看老板们列出来的意见。

- “我们就不能弄些好看的按钮放上去吗？为什么不用那种闪亮、光滑的按钮？”
- “我想要 Logo 有倒影，就像那些流行的 Web 2.0 网站一样。”
- “这配色太素了，我们需要那种可以吸引人注意的颜色。”
- “还要漂亮一点的表单，现在这种表单看起来太像那些难看的程序了。”
- “我还希望页面看起来更……有趣一些，你能明白吗？”
- “首页上要有食物的图片，这样可以让人们有食欲。”
- “我非常喜欢亚马逊那样的，你们能把首页做成那种样子吗？但是不要那种标签似的导航和杂乱无章的内容。这个要求应该不难吧？”

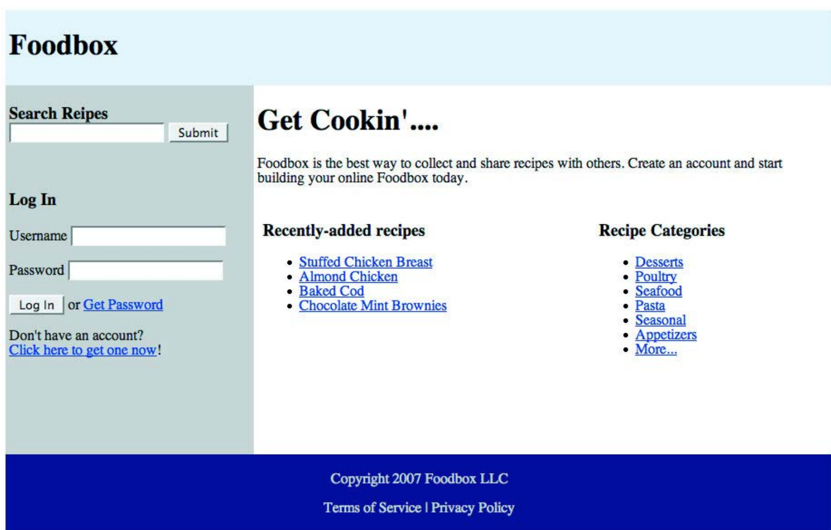
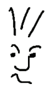


图 2-1　老板们认为这个页面太平淡了，我们会在本书中慢慢改进这个设计

在上面的这个列表里面，那些负责签支票的老板们列出了很多看起来很奇怪的要求，而你的任务就是设计出一个能让他们满意的网页。



小乔爱问……

在哪里可以看到 Foodbox 的页面

去这个网址看看：http://yourfoodbox.com。你会发现我们的目的是用一个简单、直接的设计来演示本书所要讲到的技巧。或许这个设计并不会讨每个读者的欢心，但是它确实是最适合初学者实现的简单设计。

在设计的世界里，你需要牢记的是：众口难调。你将面临的真正挑战是对书中知识的理解和应用。这本书只是一个指导，具体到字体、配色和其他设计的时候，还得看你自己。

最后要说的是，如果你看上了一个域名，尽早买下来。正如你看到的，我们想用的 http://www.foodbox.com 这个域名已经被人用了，是另外一个站点。我们只能用 http://www.yourfoodbox.com 这个域名了。通常来讲，如果你是一个域名的第一个买家，价格会很便宜，但是一个经过转手的域名则会变得很贵。

那么要从哪里开始呢？首先需要做的是了解客户想从你正在设计的这个网站得到什么。你拿到的这些反馈意见是个很好的切入点，同时这些列出来的要求也说明了上一次的设计中对客户要求的理解还是不够深入。在开发过程中，收集客户的需求和设计页面这两个步骤一样重要。你要

做的是运用你的经验，针对客户的问题提供解决方案。

然后你需要理解设计的网站到底是用来干什么的，还需要把握这个网站的目标用户。不同的用户会对不同的网站有不同的心理预期和操作习惯。因此，搞清楚你的目标用户是哪些，看看你的竞争对手是怎么做的，扬长避短。这样，当你面对客户的时候，就可以适当地提出这样的问题：“你们有没有这样想过？”

最后一点，一旦拿到了客户的需求，开始对之加以分析的时候，你就可以开始画草图了（参见图 2-2）。对，我指的是用纸和笔打草稿，这样做的原因我马上就会解释，但是首先让我们来看看怎么从客户给的反馈中提取我们想要的信息。



图 2-2　第一张草图，一个除了 Logo 就几乎没有图片的页面

客户很难对付，但是也别对他们太苛责了

的确，那些人可能会提出一些奇怪的要求，很难对付。但是你需要记住的是，这些人让你的专业知识有了发挥的机会，你的工作就是理出他们的真正需求。他们虽然已经尽力了，但还是不太会清楚地表达自己的想法。而你则需要根据自己积累的经验，从他们的含糊表达中找到重点，从而深入了解他们的困惑。

很多开发者都认为客户并不知道自己的真正需求，但是我觉得他们只是不会表达而已。其实，他们只有在看过一些不满意的设计之后，才会比较清楚地意识到他们的需求。因此，最好的方式就是保持和客户的沟通，不断给他们看你的设计方案，这样他们才能告诉你方向是否正确。保持沟通顺畅是网页设计很重要的一点，当然，在开发程序的时候也是一样。

2.2 收集需求

重新设计一个已经存在的应用程序时，你需要清楚地了解应用程序的功能，需要跟老板和用户谈一谈。你也应该读一读源代码，试用一下原来的程序，或许还会看看竞争对手的软件是怎么做的。重新设计网站也需要这一步骤。

任何一个项目都是从收集需求开始的。我们现在的任务就是回头看看 Steve 扔给你的那些笔记（参看 2.1 节）。你要从最基本的要求入手，逐步改进网页。

你会发现自己需要学习怎么去做按钮和其他一些图像。页面上有些链接需要用按钮来代替，同时表单中的按钮也应该更新。

需要注意的是，在设计的时候不要一味赶时髦，还要小心翼翼地平衡客户的要求。倒影和图片非常流行，你的客户也提出了相关的要求，所以你要学会做出这种效果——在 Photoshop 里面是很容易实现的，同时你还需要做出一个数字版的网站 Logo，从中你能够学会怎样制作 SVG（Scalable Vector Graphic，可伸缩矢量图）图像。

关于配色的要求意味着你需要了解一些基本的色彩理论，要了解如何挑选一种适合页面的颜色。当然，你也可以用图片、配色和 CSS 技巧来让页面或者 Web 应用的界面看起来温和一些，这会打消客户关于表单外观的顾虑。

另外，这是一个有关食品的页面，不可避免地要用到不少食物的图片。而且，那些竞争对手的页面之所以受欢迎，图片也有一部分功劳——它们能唤起用户的食欲。你找到一些图片之后，可能会需要对它们做一些调整，使之适应你的页面。这就涉及图像处理的一些技巧了，例如图片的修饰、亮度的调整和重新取样，等等。

还有一些要求可能表达得不那么清楚，甚至令人费解。比如说“让页面看起来更有趣”，我的建议是千万别把它太当真。我遇到过很多类似的情况，大多数时候，在经历了硬着头皮蛮干、不断试验和出错之后，外加一点小运气，你就可以达到这个标准。只要好好满足了其他方面的条件，你的页面也不会差到哪里去。

更糟糕的情况是，有时候客户会让你尽量模仿一个已经存在的网站的页面样式，然后把内容放上去就好了。但这个要求还是能带来一些有用的信息的。回头看看那个列表，你会发现最后一个发话的老板提出的意见简直让人摸不着头脑。所以干脆就别想了——这个说法可能看上去不怎样，但是那种糟糕的建议就该被忽略。遵守良好的设计准则，跟你的客户保持畅通、及时交流，那些看起来不可理喻的要求最后会慢慢得到纠正。

2.3 明确目的

设计这个网站时，要时刻关注一条：服务于你的目标用户。一个不错的主意是去问问你的客户他们平时喜欢去哪些网站，将它们当做参考。并不是说要将这些网站作为例子照搬，但是了解这类网站有助于你确定客户的口味。通常，客户会关注那些竞争对手的网页是怎么做的，但有些人会试图借鉴那些完全不相干领域里网站的设计。人们总是会说“照着 eBay 的做就好了”这种话。客户之所以这样说，是因为他们对这类网站比较熟悉。

在进行 Foodbox 设计工作的时候，要记住你是在为客户和网站目标用户而设计。这工作不是为你增添炫耀资本的，不要企图用些刚刚学到的很炫的技术来震撼你的同事。客户和他们的目标用户才是第一位的。

专注于目标客户

几年前我的一个客户要我帮他重新设计一个大概有 100 多个页面的网站。他希望新网站能帮他更有效率地出售自己的服务。原来的那一版是他的一个朋友设计的，几个盗版的图片，一些会动的图标，几种毫不起眼的颜色嵌在黑色的背景上，还有一小段 JavaScript 代码，作用是让公司的电话跟在鼠标后面飘动。

这个客户的生意口碑不错，只是他的网站相对比较寒碜。我的第一版设计很快就被他否定了，原因是“不够有趣”。他总是让我去看一些他喜欢的电台网站，因此我不得不跟他解释这是完全不同的两个市场。经过漫长的协商、小心翼翼地说服和相互妥协，最终我们完成了一个很不错的网站，让他的公司感觉上提升了几个档次。两年内，公司业绩翻了几番，他也不断地对我表示感谢，谢谢我帮助他把握了正确的方向。

我想要强调的是，设计时最需要把握的是目标用户和制作这个网站的真实目的。你可能会需要作出一点妥协，但最终的结果往往会更好。

要确保所有人都能看出网站的用途。它是用来展示信息的，还是用来激发顾客的购买欲的？是用来娱乐用户的，还是用来收集数据的？具体说来，设计一个展示暑期大片的网站跟设计一个在线零售商的网站肯定是不同的。

要尽可能了解网站的用户，要做到对一系列问题心中有底。网站是为那些零散的用户偶尔访问准备的，还是为那些专业用户的日常工作准备的？了解网站用户对你的规划设计很有帮助。比如，为小朋友设计的网站和为房地产公司设计的网站就是两码事。

2.4 从哪里入手

你已经知道了客户的要求，也对正在建设的网站有了充分的理解，现在是开始行动的时候了。如果你把那些要求按照逻辑顺序排列一下，它们会以这样的方式呈现。

(1) 画出一些基础的设计草图，并且争取其中一张能够得到认可。

(2) 挑选配色。

(3) 挑选字体。

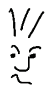
(4) 在 Photoshop 里实现基本的设计。

(5) 为 Banner、按钮和其他元素创作图片。

(6) 制作 HTML 和 CSS 模板。

(7) 测试设计的兼容性和可访问性。

接下来，本书就会带着你完成上述每一个步骤，教会你完成这些要求所用到的技术和理论。



小乔爱问……

为什么不能直接在 Photoshop 里开始做图或者用 HTML 写页面样式图呢

对于创作来讲，笔和纸是非常重要的工具。更重要的是，画画的速度比在电脑上操作的速度快多了。并且，画出来的结果可以随时丢弃——反正没有为它投入很多精力。

如果你是程序员，肯定用过办公室里的白板，肯定在上面画过一些简单的图表来和团队成员交流。你跟客户交流的过程中也可以使用这个方法，因为一个不懂技术的客户可能会被你在笔记本电脑上敲打单击的动作打断思路。相比较而言，铅笔和纸则是很好的交流方式。当着客户的面画出一些草图，然后直接给他们看，让他们谈谈想法。

这样做的目的是加深团队和客户之间的交流。你的最终设计可能会跟初稿差十万八千里，但是做过设计师的人都知道这再正常不过的了。是花几个小时在电脑上做草图，还是花几分钟用笔和纸画一个草图？纸笔是你团队中的一员，让它们帮助你碰撞出思维火花吧。

2.5 画出你的想法

你应该快速地画出你的想法，以便跟别人分享，进而做出修改。有时你甚至都可以从客户那里得到建议。

现在，去拿一支笔和一张纸来。听话。

准备好了？那开始吧。

画设计草图的时候，要做到对页面包含的内容心中有数。哪些链接是需要放在首页上的？哪些元素是首页必须包括的？图 2-1 中包含了以下几个元素：

- 网站名字；
- 搜索字段；
- 登录表单；
- 一段关于网站的简介；
- 最新的菜谱列表；
- 分类列表。

除此之外，主页还可以包含一些信息页面的链接，例如：

- 网站条款；
- 注册；
- 隐私条款；
- 联系方式。

现在让我们一起画一些草图吧（参见图 2-3）。

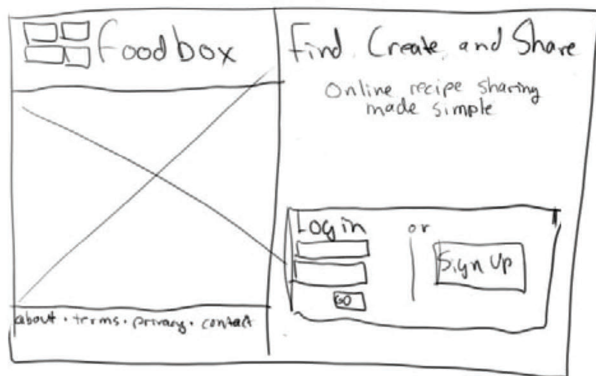


图 2-3 第二张草图，多加了一些图片。相对于第一张草图，最大的区别在于这张图在左侧留下了一个较大的空间，可以用来放一张有意思的图片

2.5.1 一些约定俗成的布局风格

你大概已经注意到了，不少页面都有一些共有的特点。比如说大多数页面都在抬头有一块地方显示网站的名字或者 Logo。大多数网站都会用分栏的方式来组织内容，其中至少有一栏是用来显示导航或其他内容的侧边栏——通常网站都会有顶部和左边栏两个导航区域。几乎所有的页面都会有一个页脚，用来显示版权信息和附加的链接。

之所以网站之间有这么多相似的地方，是因为开发人员会模仿已经成功的例子。例如，所有的新闻页看起来都差不多，这不是巧合。毕竟，所有的报纸的布局也是相似的。

日积月累，用户就会对页面的共性有所期待。一个合格的页面应该做到让用户能够快速找到他想要找到的内容，而不是在页面里茫然四顾或在网站中点来点去。页面需要一个很简单的浏览方式，本书会帮你慢慢达到这个目标。反过来说，如果你没有按照约定俗成的规范来设计，就会让网站用户感到困惑。

画草图之前可以浏览一些网站来寻找灵感，可以找类似领域的网站来看看，也可以参考一些完全没有关系的网站来对比，在此过程中或许可以发现你的竞争对手缺乏的元素，并在设计中把这些元素变成你的优势。记住最基本的原则：设计一个可以有效传播信息，同时用户又很熟悉的版式。

2.5.2 三张草图

每个项目你都需要为客户准备三个不同的设计。一个简单、保守的，一个复杂的，一个中庸的（既有点保守的元素，又有些出挑的设计）。

即便你不是一个好的设计师也不用太过担心，草图不需要很漂亮。它们的主要作用是把你的想法呈现出来让别人可以看到。

现在就让我们来看看我根据现有的要求攒出来的三张草图。第一张最简约，但不一定很好看（参见图 2-2）。页面上除了登录框和注册按钮之外，并没有其他太多的东西。这个设计的文字比较多，因此可能需要不同的配色、渐变和阴影来区分不同的区域。文字比较多的好处是提高页面在搜索引擎中的排名，缺点则是看起来不太有趣。

第二张草图（参见图 2-3）展现了一个多图的设计，在左侧为大图片留下了空间，把登录和注册的区域放在了右边。这个页面相比第一张图更吸引人些，但问题在于它没有展示足够多的吸引用户深入访问全站点的信息。

第三张设计图则具备了更多的功能，整合了现有页面中的元素，同时将站点的分类变成了标

签云（参见图 2-4）。这个设计保留了搜索框和其他一些链接，但是移除了登录注册区域，只留下了两个按钮。虽然跟原始设计有点类似，但是这一版增加了一些图片元素，还有一些留白，用来向用户解释网站的内容和使用这个网站的理由。



图 2-4　第三张草图：完善了不少功能。草图运用了现有页面的一些元素，同时也添加了一些新的东西

向客户寻求帮助

当有客户找你帮忙设计新网站时，记得做一些调查工作。问问他们喜欢的站点，让他们告诉你喜欢这些站点的原因。他们给出的答案最好是："我喜欢 Blinksale（http://blinksale.com）的配色。"，或者是"Amazon 的标签式导航很不错！"之类的。当然，你不用照搬这些设计，但是这个过程能够让你对客户的口味有个大概的感觉。然后你就能利用这些反馈，加上自己的判断和经验，做出一些不错的设计。

当我展示设计作品的时候，我会准备一个比较保守的设计，一个比较艺术性的设计和一个兼有两者特点的设计（那种既有简单元素也有复杂设计的版本）。通常，客户会选择中庸一点的，并提出要从另外两版引入一些新的东西。当你在向客户展示设计草图或者样式页面时，你不是在展示最终的产品，而是在向客户说明自己的想法，然后让相关的探讨能够继续下去。就算客户提出了修改要求，也不要灰心。需要时刻记住的是那是客户的页面，不是你的个人网站。

千万不要只向客户展示一个方案，因为他们喜欢有选择，这样会有参与感。有些客户会向你

咨询该怎么做，但实际上应该是客户自己提出应该怎么走下去。永远不要设想客户的想法，这样会显得比较傲慢，甚至会伤害你跟客户之间的关系。

好了，现在草图已经完成了，该是见见老板们的时候了。

2.6 挑选草图

Steve 跟那些老板们开完会后，面带笑容地拿着一张草图过来了。看起来他们选择了第三张（参见图 2-4）。当然，他们希望能尽快见到带配色和图片的样式页。

其实这是一个迭代的过程

有一次我听了一个关于软件开发的讲座，主讲人是 Robert Martin，他把软件开发比作写书。首先需要有一版草稿，然后会对草稿进行几次修改，直到满意为止，这样得到的版本就是终稿了。设计也有类似的过程，只不过在完成几次修改得到最终版之后，客户很可能会讨厌你的设计。于是你需要做出让步，可能会把几种颜色改成自己讨厌但是客户喜欢的颜色。这样会有一种自己的设计被毁掉的感觉，而这种感觉会给设计师带来挫败感。作为程序员，你应该已经熟悉了这种需求驱动的项目开发过程。只要把设计当做是类似软件开发的过程就好了，它也需要不断地完善、重写和重构。

2.7 小结

网站再设计的过程归根结底就是跟客户沟通的过程。一些客户对自己的需求很清楚，但大多数都需要在你的帮助下才能理清思路。有条不紊地对客户提出一些疑问，并耐心地倾听，最后总会有一个好的再设计方案出台。

接下来，客户们就需要看到带配色的页面样式图了，接下来你就要开始学习一些配色和选择字体的技巧。这些技巧会帮助你在下一次碰头会之前做出来一份好看的页面样式图。

第 3 章

配　色

在第 2 章我们绘制了网站草图，目标是完成一个样式设计，那么接下来要做的就是挑选颜色，并建立我们自己的配色模板。

网站设计往往是成也配色败也配色，这其中的成败取决于色彩的使用和搭配。配色可以调动用户的情感，可以在细微处吸引用户的注意力。这一章是本书的重点，因为这些内容是创造动人网站必不可少的知识。

伟大的设计师似乎对颜色有天生的敏感，往往是直觉和经验引领他们去为页面创建配色模板的。那些配色模板（或者叫颜色组合）是基于实践而得出的，这有点像开发人员经常遇到的设计模式。了解了颜色之间的关系之后，你就能挑出那些看起来很搭调的颜色，这跟为网络应用选择适当的设计模式一样容易。

3.1　色彩基础

在我们日常生活的这个立体的世界里，物体会吸收和反射不同的光线，而我们的眼睛，就将接受的这些反射光理解为颜色。人们通常用三种方式来表述一个颜色：颜色的名字[①]，颜色的饱和度和颜色的亮度。

当需要处理颜色的时候，你不得不考虑很多东西。关于色彩的遮蔽效果，哪种色彩占多少分量，哪两种色彩放在一起才看着顺眼，另外还要想想用户会怎么去解读你敲定的配色等问题。在这个部分，你会了解到这所有的东西是怎样运作的。

3.1.1　色调、饱和度和亮度

当人们谈论一个物体的颜色时，他们通常是在说色调。实际上你一直在使用色调这个概念，

---

① 即色调。——译者注

比如在商店买香蕉（绿的是没熟的），开车时试图趁黄灯快速经过路口。

*饱和度*指的是颜色在图像中的分量——高饱和度的色彩更加绚丽，而低饱和度的色彩则会发灰，显得沉闷。如果你降低某种颜色的饱和度，这种颜色就呈现一种*水洗*的效果。某些时候，这反而是个好事——这种色彩可以用来让那些锐利具有冲击力的色彩边缘变得柔和。

调节色彩*亮度*会让颜色变亮或者变暗。想想在咖啡里加入牛奶的情况：一杯棕色的咖啡会随着牛奶的添加而变成亮棕色或者暗棕色。

改变颜色的亮度和饱和度会让颜色的外观发生变化（参见图 3-1）。

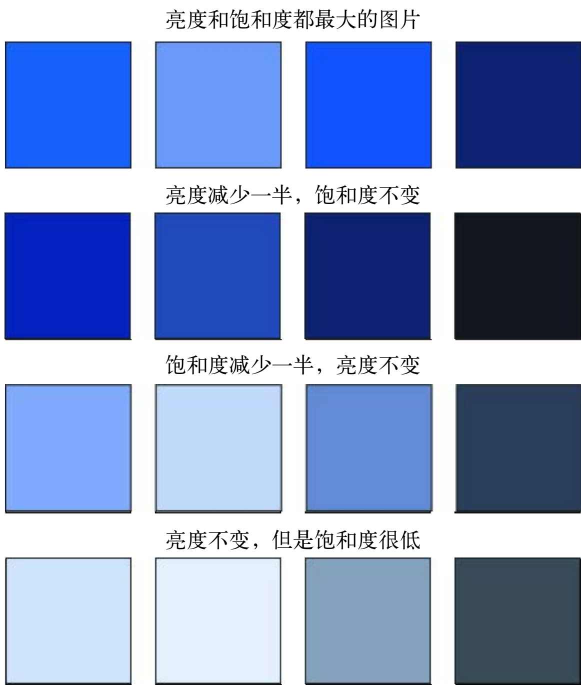


图 3-1　亮度和饱和度（另见彩插）

3.1.2　加法混色和减法混色

你看到的颜色可能跟打印出来的颜色不同。本质上来说，打印在纸上的颜色和自然中的颜色是有区别的。在自然界中，光线是被反射的；而在屏幕上，光线是被投射的。在屏幕上，颜色的混合方式是*加法混色*；而在印刷中则是*减法混色*。如果你拿一幅画中的颜色和电脑屏幕上的颜色作比较，就能很清楚地看到其中的区别了。

当你用颜料、蜡笔或者马克笔涂鸦时，所面对的三原色是黄色、蓝色和红色。整个过程其实是这样的：你从白色开始（所有光混合的结果），然后滤掉你所不需要的其他颜色，从而得到实际想要画出的颜色。比如你用红色蜡笔画画时，所做的其实是让除了红色的其他颜色都被吸收掉（减掉），这样就只有红色反射到人眼中了。

从颜料混合的过程中可以观察到减法混色。你大概已经知道，黄色和蓝色混合会得到绿色；红色和蓝色混合得到的是紫色。当你把所有的颜色都混合起来的时候，就只能得到黑色了——因为物体会把所有的可见光都吸收，然后就没有光线能反射到人眼中了。实际上，香蕉是没有颜色的，它没有能量去产生有颜色的光。香蕉只能反射黄色的光，同时吸收所有其他颜色的光，所以它看起来是黄色的。

电脑屏幕的显色原理则是基于加法混色的。在这个系统中的三原色是红色、绿色和蓝色。这些颜色被混合投射出来，变成光线。与香蕉不同的是，电脑屏幕上的图像并没有反射光线，而是确确实实地发射出了光线。屏幕上的颜色是从无到有的——从黑色的显示屏开始，慢慢地增加颜色。当将红、绿、蓝三色混合起来的时候，就得到了白色；如果没有混合任何颜色，就呈现出黑色。这个过程就是“加法混色”，你的眼睛接受的光线是屏幕发射出来的。在这种情况下，将绿色和红色混合会得到黄色。[①]

好，那这些东西跟网站设计有什么关系呢？其实了解不同的色彩模式对你来说很重要。当你在电脑上做跟颜色有关的事情时，你可以选择用 RGB 模式，也就是加法混色；或者 CMYK，减法混色，其中 C 代表青色（cyan），M 代表品红（magenta），Y 代表黄色（yellow），K 则代表关键色（key，通常是黑色）。当你设计网站时，一定要用 RGB。但是，如果你打算将做的东西打印出来，那么就应该选择 CMYK，这种色彩模式被大多数的四色打印系统所采纳。

3.2 色彩环境感知

仔细观察图 3-2。左边的蓝色长方形看起来比右边那个要暗，但实际上两个长方形的颜色是一模一样的。这种会欺骗你眼睛的现象叫做*色彩环境感知*，这种现象有时会让人很伤脑筋。

有一次我为一个客户更新他的公司主页。客户要求将一些红色的字母放在浅蓝色背景的 Banner 上面，而且他希望我用的红色要跟页面上的其他红色一模一样。

---

① 有时“加法“和“减法”混色的说法会让人感到困惑。在这里，我的标准是颜色的反射——以色素为基础的颜色混合的相减过程。当我为了得到绿色而把黄色和蓝色混合起来时，看上去我是在加入一种颜色，但实际上我是减少了某一个种类的可见光，因此这个过程不是“加法”。

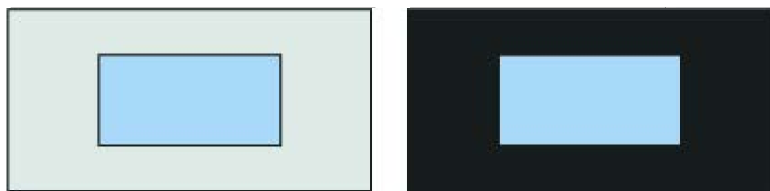


图 3-2　哪个蓝色方块看起来更暗（另见彩插）

作为一个开发者，你应该能看出问题在哪儿。顾客告诉了我们实现的效果——他想让 Banner 中的红色跟页面上的其他红色看起来一样，而我却需要找一种较为亮一点的红色才能达到这个效果。

实际上，当我用客户指定的红色时，他并不喜欢那个效果，那种红色看起来不对劲。当我把这个红色调得稍微亮了一点之后，它看起来就跟页面上其他红色差不多了，客户也很满意。

色彩所处的环境在很大程度上会影响它在页面中所呈现出的效果。即便从技术角度来看，颜色选择正确无误，也可能还需要额外的调整，才能让那些颜色看起来是正常的。

Fluting 是产生这种效果的原因。Fluting 指的是你的眼睛会把相近的颜色进行混合。你可以好好利用 Fluting（参见图 3-3）。通过例子可以发现，Fluting 可以实现渐变。如果过渡不够平滑，则会出现带状的效果。但是如果过渡是通过一些很小的变化来实现的，那么你的眼睛就会忽略那些变化，把所有颜色都混合在一起。



用较少的色块来显示一个颜色到另外一个颜色的渐变，过渡很明显



如果增加色块的数量，我们的感官就开始自动地融合这些颜色



成千上万的色块则让颜色的过渡平滑无缝

图 3-3　色彩 Fluting（另见彩插）

3.3 用颜色唤起情感

从小我们就知道，颜色跟情感、情绪和感觉有关。对网站进行配色的时候，要仔细考虑你的选择会带来的不同效果，这很重要。如果用错了红色或者蓝色就可能给用户带来不愉快的感觉，或是造成困惑。

3.3.1 暖色

顾名思义，暖色会让你联想到温暖，阳光和热。有人认为看着暖色调的颜色就会感觉到温暖。

1. 红色

红色是一种浓烈的颜色，代表着爱情、愉悦、幸福和浪漫。它有时也会用来表现情欲、愤怒、战争、紧急或者危险。在网站里面，红色总被用来显示警告或者错误信息，它能很快地吸引用户注意。

2. 黄色

用户很难把注意力集中到黄色上，但是如果运用得当，这种颜色会让人联想到智慧和幸福感。很多应用都用渐渐消失的黄色来让用户知道他们的操作成功了。

3. 橙色

橙色可以像黄色那样让人振奋，但是也会给人一种傲慢的感觉，这取决于橙色偏红的程度。有专家认为，橙色中的红色元素对大脑有刺激作用。

3.3.2 冷色

冷色给人带来清凉和沉着的感觉。它们看着很舒服，你可以用这些颜色来让页面变得缓和。冷色系包括了蓝色、绿色和紫色。

1. 蓝色

蓝色代表冷静、舒缓和凉爽。当蓝色被稀释后，它可以让用户感觉更放松。但是如果蓝色变得很深，则会引起悲伤和沮丧。

2. 绿色

人们喜欢将绿色与自然、希望、健康以及有同情心联系在一起。然而，如果运用不当，绿色还可以引起嫉妒（可能这种印象出自于英文表达法“green with envy”）。除了嫉妒，绿色还可能

代表着贪婪、罪恶和混乱。一些绿色可以让人眼得到休息，这些绿色可以让用户变得和缓。但是不正确地使用绿色则会给用户带去不愉快的感觉。

3. 紫色

紫色是众多不会经常在自然界中出现的颜色中的一种。你也许曾经见过紫色的花瓣，但是大部分紫色的东西还是人们创造出来的。紫色总是和尊贵与神秘联系在一起，主要原因是古时候制造紫色是一件极为困难的事情。紫色是红色和蓝色的混合，这意味着它兼有两者的一些特性。淡紫色总是让人想起自然、和平、平静和精神力。暗紫色则会唤起沮丧的感情。大量的紫色可能会让人看起来不太舒服。

3.3.3 中性色

黑色、白色、银色、灰色、米色和棕色是一类颜色，它们填补了冷色和暖色之间的空白，使用它们作为背景色可以突出其他的颜色。

1. 黑色

黑色呈现出威信和优雅，使用得当的话，黑色会很有冲击力。然而，黑色也跟忧伤、死亡、绝望和多虑联系在一起。如果要在设计中使用黑色，你必须仔细考量目标用户。

2. 白色

白色代表纯洁和完美，最适合干净的网站。但是如果使用过多，也会显得无趣和没有创造力。另外，白色作为背景色的时候可以让其他颜色更突出。

黑与白：不是颜色的颜色

从技术上来说，黑与白不算是颜色。当我们说到屏幕上的颜色的时候，黑色代表的是所有颜色的缺失，白色则是所有颜色的混合。回忆一下有关加法混合和减法混合的概念，我们可以知道，绘画时则和上述情况正好相反。

但是，虽然它们从技术角度来看不是颜色，但是它们仍然能唤起人们的某些情绪，因此我们还是要将黑和白当作颜色来处理。

3. 棕色

棕色会激发饥饿感，也会让人感觉健康和简洁。另外，也有人觉得棕色不干净、很脏。这种效果当然不是你的网站所追求的。

4. 米色

米色可以让人放松。它是一种保守的颜色，来自棕色和白色的混合。米色很适合做背景颜色，因为它显得很冷静，能突出其他的颜色。

5. 灰色

灰色很少能唤起人的什么情绪，就算有，也大多是忧伤、悲哀和沮丧等。灰色给我们的感觉跟阴天带给我们的感觉差不多，在色谱上靠近冷色。

灰色也是个很有意思的颜色，如果把它弄暗一点，它就接近黑色的效果；如果把它变亮，则会呈现出白色的特点。

3.3.4 颜色和用户

要记住，选择的颜色对用户情感的影响，有时这种影响取决于用户个人的偏见。这种偏见也许是由于某些不好的经历或者记忆而产生的，但大多数时候都跟文化有关。

以红色为例，也许我们总觉得红色代表了情欲、愤怒或者激情，但是在中国，这种颜色是运气和喜庆的代表。在印度的某些地方，红色则代表着功成名就。在南非，红色则是一种哀伤的颜色。在西方世界黑色代表着悲哀，但是中国人有时会用黑色来呈现高品质。

Brandcurve 写过一篇关于不同文化中颜色的不同含义的文章[①]，很不错。如果你的网站需要面对国际化的用户，要记得除了文字之外，还要给颜色做本地化。

3.4 配色方案

不同的颜色组合起来，有的好看，有的不好看。配色方案其实就是一组组颜色的组合，这些组合能够在视觉上吸引观众。让我们随意看看几种不同的配色方案，同时探索一下它们的用法。

在选用一种配色方案之前，你需要了解一定的色彩理论，最好的方法就是观察一个色轮。色轮可以展示不同颜色之间的不同关系。我已经画了一个很简单的 RYB 色轮，或者叫混合色轮，这种色轮以红色、黄色和蓝色作为主色（参见图 3-4）。在这一章中，我会用混合色轮来展示几种不同的配色方案。

① http://www.brandcurve.com/color-meanings-around-the-world/。

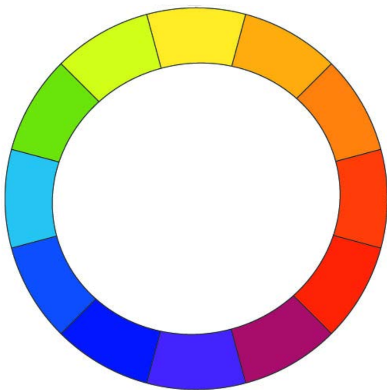


图 3-4 在混合色轮上，主要的颜色有红色、黄色和蓝色。这种色轮也经常称作 RYB 色轮（另见彩插）

3.4.1 单色方案

单色方案只会用到一种色调（参见图 3-5）。先选定一种色调，然后再调整色调的亮度和饱和度，得到不同的变化，组合起来就成了单色方案。

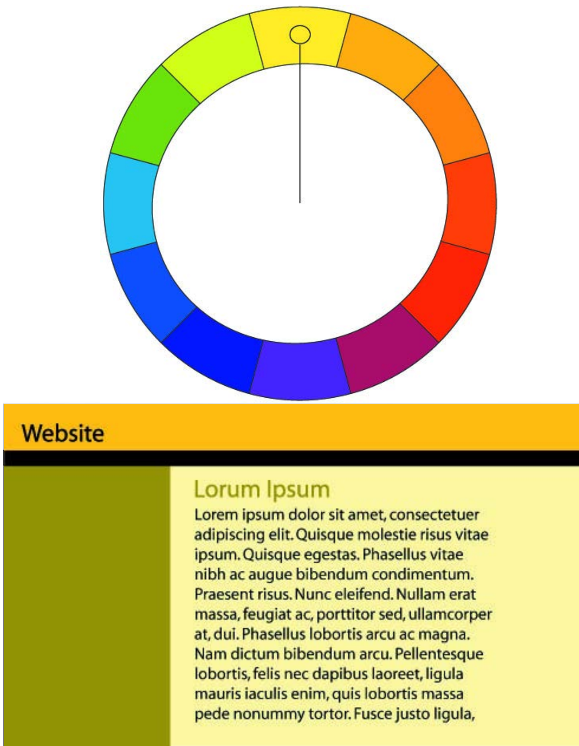


图 3-5 单色方案（另见彩插）

这种方案给设计带来内涵和深度。当你在页面中使用这种方案的时候，页面上的其他元素（照片、图标等）就会显得很突出。而且，制作单色方案非常简单，它最适合那种内容至上的站点。

3.4.2 相似色方案

在混合色轮上，一种颜色两边的颜色就是这种颜色的相似色。将这三种颜色（基础色和它的两个相似色）组合起来就成为相似色方案（参见图3-6）。这种方案高度可控，同时色彩的临近感也会加强整个方案的效果。

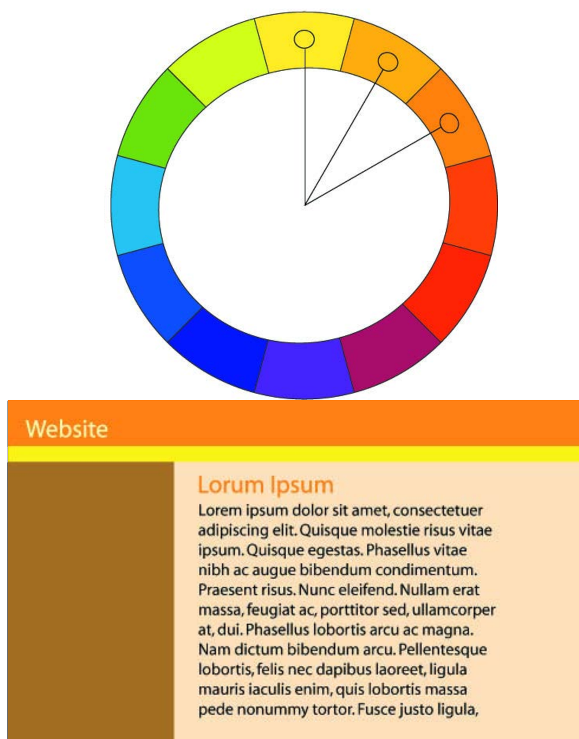


图3-6 相似色配色方案（另见彩插）

相似色方案需要在色轮上选取相互关联的颜色，一种颜色作为主色，另外两个跟主色类似的颜色则起到了点缀加强的作用。

创造这种方案的容易程度和创造单色方案差不多，但是出来的效果会更好一些，因为你用到了不同的颜色，而不是一种色调的不同变化。两个附加色能够更好地衬托主色，将用户的眼球吸引到重要的内容上。

使用相似配色方案的一个主要问题是缺乏颜色对比，特别是跟互补色配色方案相比较，相似色方案的色彩对比就显得不是很明显。但这种方案确实是最适合初学者的方案，创造一个这种方案不难，然而它却可以给你一个保险的色彩范围，不会发生色彩冲突。

3.4.3　互补色方案

互补色方案选用的是色轮上相对的两种颜色为基准色，这两种颜色被看做是互相补充的颜色。紫色和黄色是很好的互补色方案，另外还有红色和绿色。在图 3-7 中你可以看到一个互补色配色方案的例子。

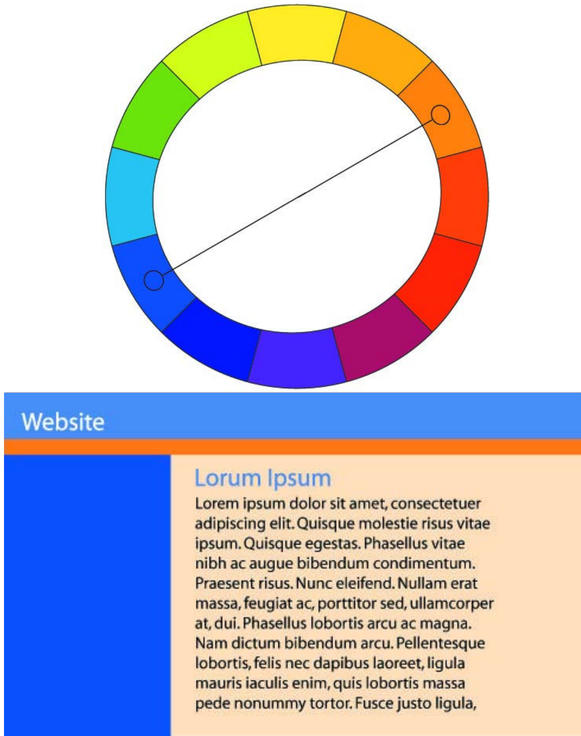


图 3-7　补充色配色方案（另见彩插）

互补方案通常难以平衡，因为选取的颜色有可能都是很亮的颜色，你需要做不少调和工作来使之看上去协调一致。有些组合，像橙色和靛青，平衡起来就非常困难。如果使用得不正确，这些难以调和的颜色就会显得突兀，甚至带给观众压力。但是，如果减少两种颜色中冷色系的饱和度，同时增大暖色系的饱和度，就可能会得到非常好的效果。利用互补色配色方案的办法是选取其中的一个颜色作为主色调，把剩下的那个作为辅助颜色来补充强调主色调。

在使用这种配色时还要小心处理文字效果。如果在一个背景色上使用的文字颜色是其互补色，可读性会非常差。这时你需要适当地调整一下饱和度。

3.4.4　分离互补色方案

这种方案更难掌握，但是很有意思。跟互补色方案类似，如果对饱和度和亮度做了适当的调整，这种配色会非常迷人（参见图 3-8）。我最推荐你尝试的也是这个方案，因为使用这种方案的设计很少，这意味着使用它很有可能让你的设计与众不同。

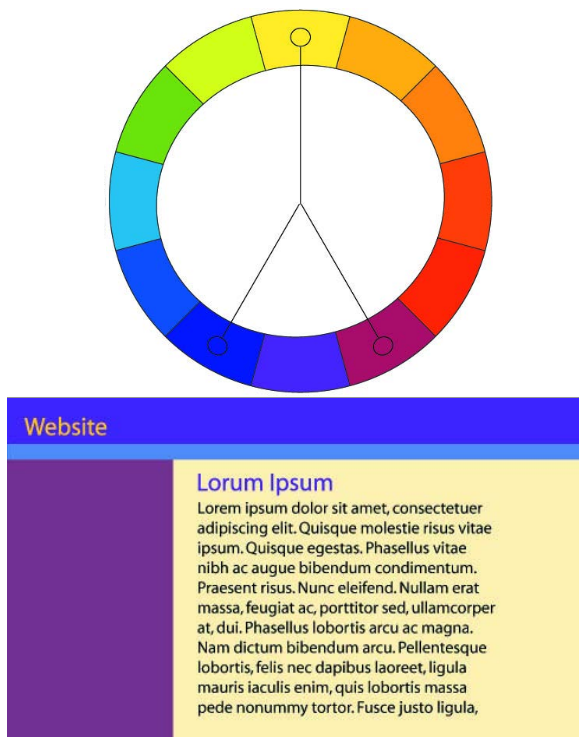


图 3-8　分离互补色方案（另见彩插）

这种方案指的是在色轮上选择与主色的互补色相邻的两个颜色，而不是直接选取某个颜色的互补色来做设计。

这个方法除了可以让配色具有更强烈的对比之外，还能增加色彩的多样性。最终你会得到一个跟互补色方案相比更有亲和力，而且不那么极端的配色。

需要注意的是应该避免使用那些很无趣的颜色，它们会影响配色的整体效果。

你最好将这几种不同的配色方案都尝试一遍，感受它们的效果。我最爱用单色方案，因为我喜欢那种让图片突出的效果。你也可以发掘你自己的最爱，最重要的是要了解每个方案的优缺点。

3.5 网络安全色

网络安全色调色板只提供了216种颜色，这些颜色在不同的操作系统上都是一样的。这个概念起源于计算机显卡技术不够强大的时代。Mac[①]机上的设计师想确保对于同一张图片，PC[②]机用户也能看到同样的效果。然而，这个调色板非常单调，限制太多，上面只有蓝、绿、红各六种色调的不同组合。

现在设计师已经放弃了这种网络安全色调色板，因为大多数用户的显示器都能显示上百万种颜色了。虽然不同的机器上的色彩显示还是有区别，但是这种区别基本上可以忽略不计了。

少数经验丰富的网页设计师和一些机构仍然执着于网络安全色，另外有些图形图像程序也提供了一些选项，可以让你只在网络安全色范围内工作。图3-9展示了Photoshop的取色器（注意左下角的Only Web Colors选框）。不管你是自愿还是不得不使用网络安全色以外的颜色，都应该记住这种颜色可能在不同的地方有所差别。

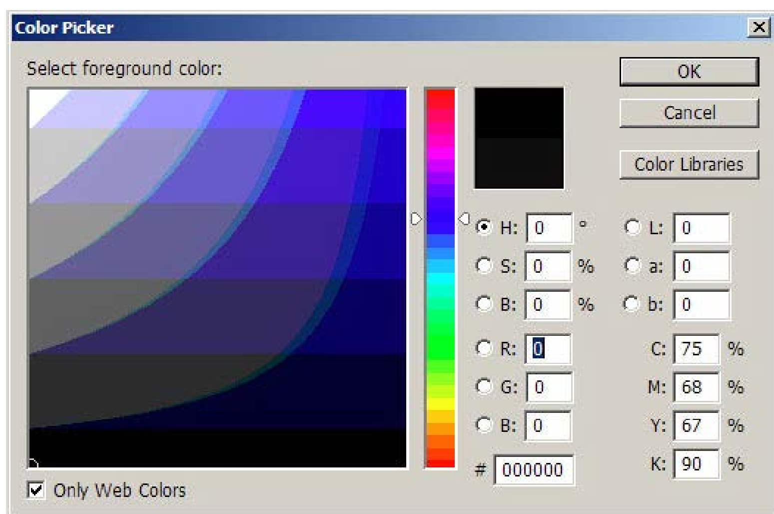


图3-9 Photoshop取色器提供了网络安全色的选项（另见彩插）

① 指苹果公司出的麦塔金系列电脑。——译者注
② 通常指安装微软公司的Windows系统的个人电脑。——译者注

三色原则

我的祖父曾经做了多年的男装生意。我很小的时候，他告诉了我“三色原理”的概念。祖父的意思是男装应该使用三种颜色：首先是两种可以互相补充的颜色，然后还需要另外一种完全不相干的颜色。我用个例子来说明吧。一个穿了白色T恤的男人，可以搭配一条黑色的裤子，同时还需要一条黄色领带，领带上可能还有点黑色和白色的点缀，这样整个人就会显得更有活力。

现在当我在做网站设计的时候，总会想起这个理论。我会挑一个背景色和一个前景色，同时选择另外一种颜色。第三种颜色需要有一种加强的效果，需要看起来很出挑。比如你可以用蓝色和灰色来做主色调，然后用黄色来做加强色。在页面上，你可以用不同的蓝色和黑色来确定一个对象，并使之显得尤为突出。

注意，我的意思并不是说设计里就只能用我说的这几种颜色。

3.6 创建配色方案

现在你有了一些色彩理论的概念，应该可以开始思考 Foodbox 的颜色搭配了。你也已经知道怎么用色轮来发掘色彩组合了，但我还是要介绍一下我挑选颜色时用到的技巧。

就像之前做页面设计时那样，我们需要将几个配色方案拿到老板跟前，让他们挑选。这里介绍两种选色方法：技术法和自然选择法。

3.6.1 用技术法选择颜色

技术选色法是，基于色彩理论来创造配色方案。先挑选一个基础色，然后用 3.4 节中讨论过的方案中的一种来给基础色搭配组合。再后调整亮度、饱和度和对比度，协调颜色混搭。这听起来可能比较难，但是在一些很棒的软件的帮助下，所有的过程其实比想象的简单得多。

HTML 颜色代码

HTML 中的颜色用三个十六进制数来表示。第一个数代表红色，第二个数代表绿色，第三个数代表蓝色。

以#FF0000 为例，这个代码的显示效果是红色，因为第一个数是有数值的，而后面两个数的值都是 0。每个数的值都在 0~255 之间，所以这里的红色值是 FF（最大值，最红），00（没有绿色）和 00（没有蓝色）。

这种方法其实是在用色彩理论来建立我们自己的调色板。直觉并不比算法和规则来得有效，所以如果你觉得自己的艺术感没那么强，这将会是一种好方法。当我手边没有图片来激发灵感的时候，我也会用这种方法来快速地建立一种配色方案。

写程序的时候，我们会用 IDE 之类的东西来让工作变得简单一些。当然你可以用记事本来写代码，但是面对一个大型项目的时候，用记事本写代码就会显得比较疯狂了。同样的道理，你可以以一个色轮为基础来手工打造一个配色方案，你也可以借助某些色彩工具来协助你做这件事。

在网络上你能找到很多的配色工具，我推荐 ColorSchemeDesigner.com[①]。这个网站的用户界面能帮助你快速创造一套配色方案，并且还可以用各种格式输出，其中包括 Photoshop 的调色板格式。在浏览器中打开这个网页，我们马上就要用。

1. 选择一个色轮

在图 3-10 中，左侧是一个传统的 RYB 色轮，右侧是一个加法混色（或者叫 RGB）色轮。设计配色方案的时候，你需要从中选择一个。那么，你能看出其中的区别吗？

Mac 和色彩

PC 显示器的 gamma 默认值是 2.2，但是 Mac 的屏幕一向用的 gamma 值是 1.8。这个差异让颜色在 Mac 屏幕上面显得更不饱和。但是，自从 Snow Leopard 开始，Mac 显示器的默认 gamma 值跟 PC 一样了。如果你在用 Mac OS X 的早期版本，你应该考虑将屏幕 gamma 值调成 2.2。在系统偏好设置（System Preferences）里的显示属性（Display Properties）里可以调整。即便 Mac 做出了这样的调整，你还是应该在不同的系统上测试颜色的显示效果，以确保颜色不会有饱和度过度或者不足的问题。

两个色轮中的互补色是不一样的！如果你以 RYB 色轮为基础，选择以黄色为基准色的互补色方案，黄色的互补色是紫色。但是如果用的是 RGB 色轮，那么黄色的互补色则是蓝色。这种差别是设计师和开发者要面临的烦恼之一，特别是当人们不知道这两种色轮的区别的时候，烦恼尤甚。甚至有些人把两个色轮当作一回事，那就更糟糕了。

设计师们为选择哪种色轮来开发配色方案争论不休。有人认为用 RYB 色轮创造的色彩方案更能吸引眼球，因为这个色轮是传统画家和设计师用的，所以人们对它比较熟悉。另外一些人则争辩说做网站设计的时候应该用 RGB 色轮，因为它更加准确地反应了色彩在电脑屏幕上的呈现效果。

---

① http://www.colorschemedesigner.com/。

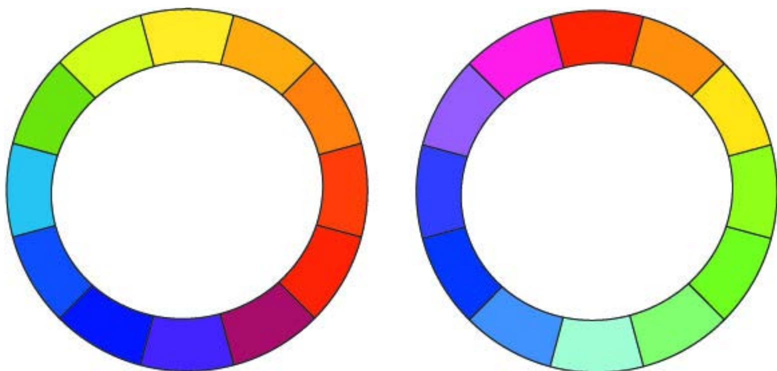


图 3-10　RYB 和 RGB 色轮（另见彩插）

那么，你应该用哪种色轮呢？比较偷懒的回答是，只要你是在电脑上做网站设计，并且是以屏幕上的颜色为准，那么用哪种都无所谓。你在本章前面看到的所有配色方案都是用 RYB 色轮设计的，但是用 RGB 色轮也可以得到很好的结果。就算是在线的配色工具用的也是不同的色轮。ColorSchemeDesigner.com 提供的取色器用的是 RYB 的，本书里我们会用这个工具来建立自己的配色方案。Adobe Kuler 使用的是 RGB 色轮。你可以随意试验，但是一旦选择了一个色轮，那么在整个网站的设计过程中就不要再变了。

2. 建立配色方案

配色的新手可能会觉得迷惘，该从哪里下手呢？你可以从 3.3 节中学到的关联性开始。思考一下，当你想到食物的时候，什么颜色会自然而然地出现在脑海中：是不是有橙色、绿色、红色和黄色。那我们就从一种经过变化的黄色作为基础色开始吧。

你可以从 ColorSchemeDesigner.com 页面左侧的色轮中随意选择一个点作为配色方案里的基础色。选择一种颜色之后，页面右侧的方块里的颜色会随之变化，方块里的预览可以让你对几种颜色的搭配有个印象。

如果已经知道了需要选用的颜色的十六进制代码，那么就可单击色轮右下角的 RGB 链接，在跳出的对话框中输入数值。如果基础色是从其他地方选择而来的——像是照片或者其他网页什么的，而你需要以这个颜色为基础快速地建立一套配色方案，那么这个功能很有用[①]。我们会用颜色#FFE500 作为基础色，这是一种橘黄色。你可以随意选择自己的方案，或者跟着我的步骤走，在对话框里输入这个十六进制代码。

① 在这里，颜色会有一点改变，因为无法将 RGB 颜色代码毫无误差地转换为 RYB 颜色。然而这个页面上的 RYB 颜色也是显示在屏幕上的，所以这种改变几乎无法被人察觉。

绿色、黄色和橙色在色轮上是相邻的。你应该已经知道，用相邻的颜色可以创建一个相似色配色方案，这个方案的一个好处是符合我们前面提到过的三色原则。在页面上把方案从单色（mono）改成相似色（analogic），这样方案就拥有不同种类的黄色、橙色和绿色，更具多样性。

色轮上的每个相邻色之间的间隔都是默认的 30°，你也可以拖动相邻色点让这个角度变小或者变大。增大角度可以让对比更加强烈。如果你一时半会儿还找不到可以抓住眼球的颜色，可以多试几次。

另外，还应该调整饱和度和亮度，看看这些调整对整个配色方案有怎样的影响。通常取色器会自己挑选一些不同的亮度和饱和度，也可以单击 Adjust Scheme（调整方案）标签，在里面自行调整。随意挪动一下滑块，试试各种设定值。记得要降低所选颜色的饱和度，还要降低亮度让它们看起来暗一点。从技术角度来看，调整亮度和饱和度后的任意色调都是可以用来放在方案里的。

从图 3-11 中可以窥得一斑。

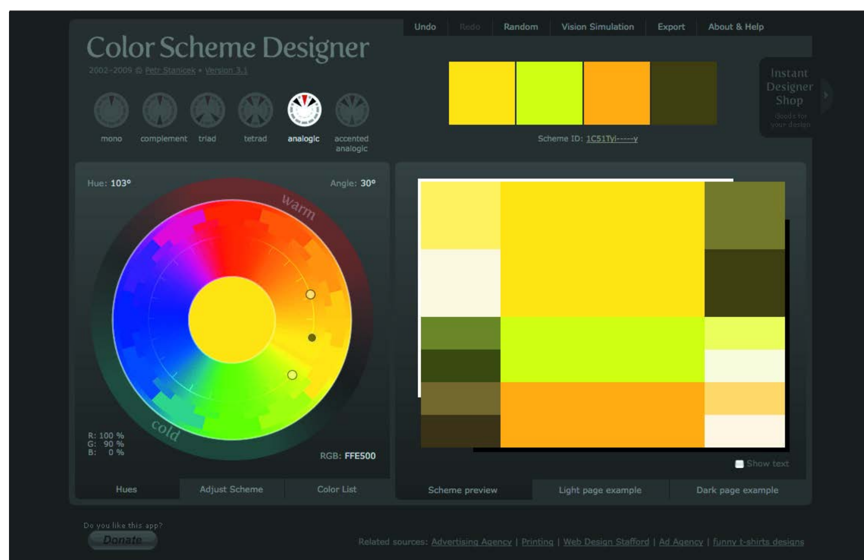


图 3-11　ColorSchemeDesigner.com 的取色器，图中是一个相似色配色方案（另见彩插）

你总会希望对方案做出一些调整，电脑自动生成的配色可能看上去不太对劲，你也不想全靠电脑来完成这个工作，就像在写应用程序的过程中，你不想依赖自动生成的代码一样。

熟悉了那些关于颜色的选项之后，可以将完成的方案存起来，下次再碰到需要为设计挑选颜色的时候，还可以拿出来参考。

选择 Export（导出）标签，挑一个导出格式，我建议用 ACO（Photoshop 调色板格式），这样，在做样式页的时候，就可以将这些配色作为选色样本了。[①]

每个方案都有一个 URL 供你下次访问。如果你想使用我为本书的例子选用的配色方案，也请访问 ColorSchemeDesigner.com[②]。当然，你也可以参考本章后面的图 3-18。

3. 用Adobe Kuler进行下一步探索

如果想用一个基于 RGB 色轮的配色软件，那么可以试试 Adobe Kuler[③]（参见图 3-12）。你也可以选择一个基础色，挑选一个方案种类，然后 Kuler 会为页面生成一个带有 5 种颜色的调色板，你可以调整调色板中每种颜色的亮度和饱和度。同样，你也可以保存这个调色板，或者把调色板跟其他人分享。

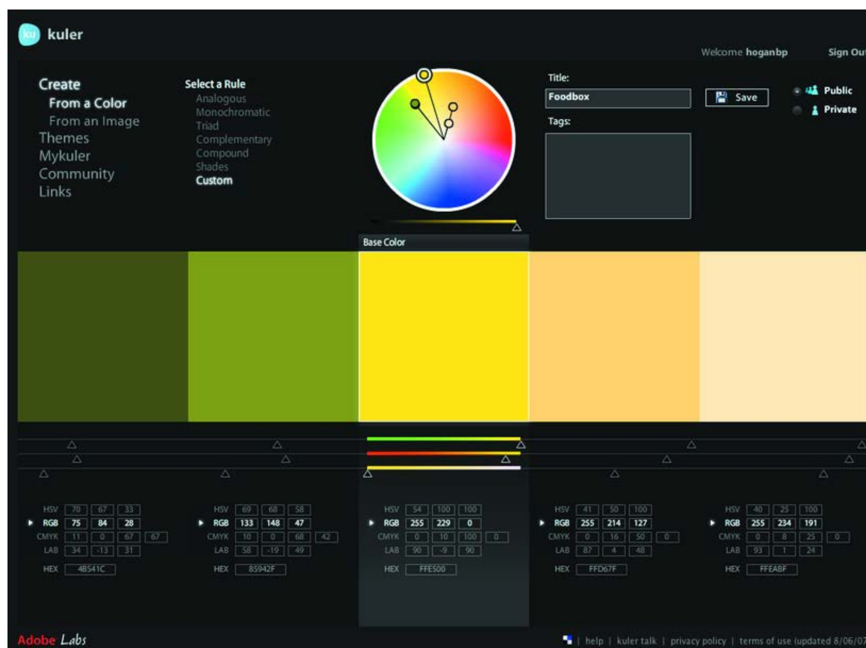


图 3-12　使用 Abobe Kuler 选取颜色（另见彩插）

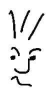
还有其他的工具可以帮助你来完成网站的配色。所以在继续到下一步之前，我们先来看看另一种选择配色方案的方法：自然选择法。

① 我使用的 Photoshop ACO 文件可以从 http://www.webdesignfordevelopers.com/colors/网站下载。
② http://colorschemedesigner.com/#1C51Tyi-----y。
③ http://kuler.adobe.com。

3.6.2 用自然选择法选择配色

色彩理论是很不错，但老是依照理论来选颜色最后也免不了觉得无聊或者太“技术”了。自然选择或自然搭配方法指的是，有一个参考物——多数情况下是照片，然后从这个参考物中选择颜色来组成自己的配色方案。这种方法是除色彩理论外另一个流行的色彩选择方法。用这种方法，可以得到很惊人的效果，但是这也取决于你选的照片或者灵感来源的好坏。当然，熟悉色彩理论也是有帮助的。根据理论，你就可以调整颜色使其达到设计方案的要求。用这种方法可能会更耗时，因为经常会从来源图片里选到不合适或者不好看的颜色。

自然选择法的最大好处就是可以跟自然合作，比如用一张食物的图片作为来源，图片里的颜色已经浑然一体，而且源于自然的颜色解读起来一点也不难。所以下次出门时，记得好好看看门外的颜色。



小乔爱问……

色盲用户怎么办

选择配色的时候，也要考虑到色盲用户，而且这一点很重要，特别是当你仅使用配色来吸引用户注意的时候。在 16.2 节中，我会谈到这个问题，你可以用那一节里所提到的技巧来测试你的配色。另外，我们用的取色软件里也有模拟多种色盲症的模式。

寻找颜色

拿起数码相机去外面走走，下面这些地方很适合探索颜色搭配。

- 花园

  去寻访大学校园、公园或者植物园，那里丰富而多彩的颜色是探寻自然配色的理想地方。

- 动物园

  去动物园拍些动物的照片吧。虎、雪豹、孔雀等一些动物比你想象中还要多姿多彩。

- 忙碌的街道

  车、交通标志和建筑都可以拍一拍。一条灰色的城市街道往往承载着丰富的颜色，比你上班走路时注意到的颜色要多得多。

这些活动有两个目的，一是强迫自己去观察周围的世界，二是让那些自然界中的颜色来激发灵感。

我们来熟悉一下这个自然选择法，你可以学习如何在设计中使用它。我从 MorgueFile[①]网站上面取了一张图。Morguefile 是一个很好的图片网站[②]，其中很多图片都可以免费使用。

看看吧，你能从这张图片里得到多少种颜色。有亮绿，草莓红，还有暗一点的黑莓色和浅一点的面包皮色。啊，从图片背景里面甚至还能得到灰色。

你可以手动将颜色都提取出来，也可以要个小聪明，用软件来帮助你做这件事。在图3-13里，你可以看到我用 Adobe Illustrater 里的 Eyedropper 工具手动选出来的颜色，但是这种方法有时候会太慢，会让人不耐烦：首先要用 Eyedropper 工具选出一个区域，然后才能在颜色选择器中找到颜色的代码值。还好，我们另有捷径。



图 3-13　从一张图片中取出的颜色［原图来自 MorgueFile（http://www.morguefile.com）］（另见彩插）

1. 用ColorSchemer Studio轻松抓取颜色

ColorSchemer Studio 有一个功能叫做 PhotoSchemer（图片配色器），这个功能可以以一张你提供的图片为基础自动创建配色方案。仔细观察上面那个草莓的图片后发现，在这张照片提供的可用颜色中，草莓上的红色太多。可能黄色或者橙色会更平易近人一些。于是我自己动手拍了一张由葡萄、芝士和胡萝卜组成的照片。葡萄的绿和芝士的奶黄色可能会带来一个更漂亮的配色方案。

---

① http://www.morguefile.com/archive/?display=111353。

② 如果最终决定在设计中使用 MorgueFile 的图片，你需要先去看看它的使用许可。网站没有单独说明你需要得到摄影者的许可，但是还是联系一下原作者为好。

去网站上把图 3-14 中的照片下载下来[①]（右键单击图像，选择“图片另存为”，然后把图片存到桌面上以便查找）。



图 3-14　我为这个项目所拍摄的照片（另见彩插）

现在准备工作做好了，你可以开始尝试 ColorSchemer Studio 中的 PhotoSchemer 功能了。

(1) 首先，开打 ColorSchemer Studio[②]，在工具栏中单击 Quick Preview（快速浏览）按钮（或者用快捷键 Ctrl+P）。

(2) 单击工具栏中的 PhotoSchemer（图片配色器）按钮（或者用快捷键 Ctrl+H），启用该功能。

(3) 单击 Open（打开）按钮，载入刚刚下载的图片，图片配色器功能会显示这张图片并提供一个有四种颜色的配色方案。

(4) 你可以把每种颜色都拖到 Quick Review（快速浏览）窗口中去，以便查看它在配色方案中的效果。你也可以增加颜色选择点的数量，最多可以达到 9 个。

同时，你还可以手动调整选择色彩。如果移动了一个颜色选择点，在颜色方案中的相应颜色也会发生改变。

(5) 试试各种颜色组合，直到找到你满意的方案。要记得用学过的颜色对比理论。你也可以看看图 3-15 中我选择的方案。

---

① http://www.webdesignfordevelopers.com/files/color/grapes_cheese_carrots.jpg。

② 目前 ColorSchemer Studio 已经更新到 2.1.0 版本，操作跟下述的步骤略有不同。——译者注

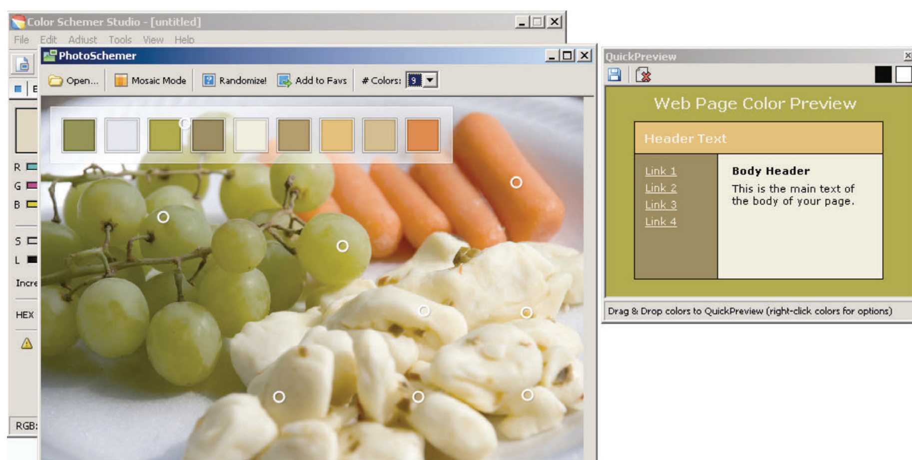


图 3-15 用 ColorSchemer 的图片配色器功能从照片中抓取颜色（另见彩插）

(6) 得到满意的颜色之后，单击 Add to Favorites（收藏）按钮，把颜色放到程序的主窗口中去（参见图 3-16）。

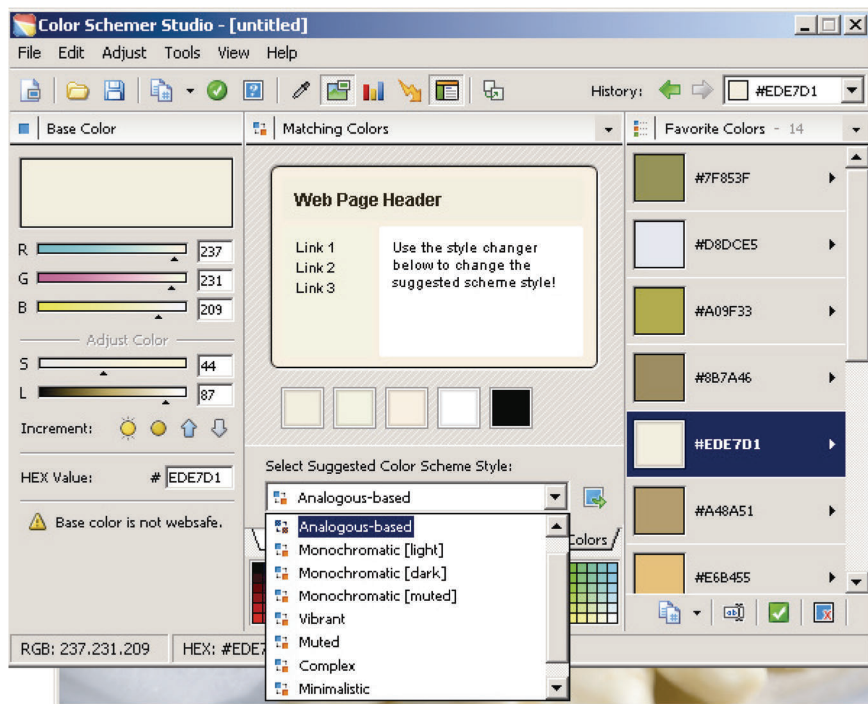


图 3-16 用 ColorSchemer Studio，以一些颜色为基础来创建配色方案。多多尝试，用任何你想要尝试的颜色来建立不同的方案（另见彩插）

(7) 选择 View（查看）> Color Wheel Mode（色轮模式），确认勾选 Computer Color Wheel（RGB，计算机色轮）选项。这样才能确保选择的颜色是合适的。

好了，现在你应该有了所有的颜色和它们的 HTML 代码了。你可以把这些代码复制到剪贴板上，也可以用 ColorSchemer Studio 再做一些配色方案。我们手头要做的只剩选择 File（文件）> Save（保存），将目前这个配色方案保存下来。

2. 将自然选择和色彩理论结合起来

你可以将这两种方法结合使用，以创造一个漂亮的配色方案。先用 ColorSchemer Studio 从图片抓取合适的颜色，用这个颜色为基础，在 ColorSchemer Studio 或者线上工具中创造配色方案。Adobe Kuler 也可以从上传的图片中提取颜色，然后调整颜色，搭配出一个方案。在图 3-17 中，你可以看到具体的功能。

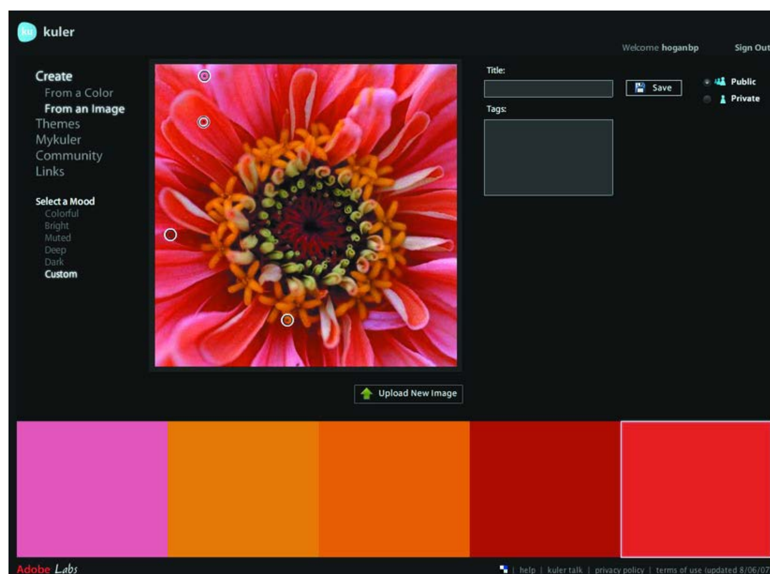


图 3-17　使用 Adobe Kuler 中的图片功能（另见彩插）

建立配色方案的过程具有很强的实验性质，可能最后得到的结果跟你选择的初始色彩大相径庭。但是只要练得越多，最后出来的效果就会越好。慢慢地，你就可以摆脱工具，凭着自己的直觉创造出一个良好的方案。

3.7　选择一个方案

我想你已经了解，通过这两种途径来为网站建立配色方案是一件多么容易的事情。现在要做

的是从这些结果中选取一个方案，以便接下来完成样式页的制作。就目前的结果来看，我觉得根据色彩理论创造出来的亮一点的配色方案最适合我们的网站。那么在本书接下来的例子中，我们会采用这一套方案。而你要考虑的，是怎么应用这一套方案。

3.7.1　前景色和背景色

你必须考虑为链接和文字选择恰当的颜色，好让它们更可读。暗色背景配明亮的前景色，反之亦然。前景色和背景色的对比越强烈，用户看起来就越容易。

要记住，你的终极目标是建立一个可用的网站，如果前景色和背景色太过相近，用户可能就跑去点后退按钮了。你在配色方面上已经花了不少时间，所以不要在这一步功亏一篑。

3.7.2　链接

链接要和页面上的其他内容有着不同的颜色，这样才能显得突出。要考虑的不仅仅是链接的颜色，还有它们的不同种类：访问过的链接、可用链接，还有鼠标划过时的链接——通常说来，鼠标划过链接的时候，链接的颜色会发生变化。

当用相似色和单色方案的时候，其实链接颜色的可选范围是受到限制的。一个比较实在的方法是用亮度和对比度的差别来区别用户已访问和未曾访问的链接。你可以将已访问过的链接变成淡一点的颜色，这样就能突出还没有被访问过的链接，只是要注意对比度的跨度，好让用户能看得出区别。

在图 3-18 中，我列出了在这个教程中使用的所有颜色。当然，如果你具有尝试精神，也可以用喜爱的颜色创造一个这样的列表。你需要记录自己在每一节里用到的颜色，这样在以后做样式页或者写样式表的时候，都能回过头去查阅。

Header #FFE500
Sidebar #FFDD7F
Main #FFF8E4
Heading #414D00
Text #000000
Links #4D3900
Visited Links #806F40
Hover Links #807940

图 3-18　在本书里我将会用到这些颜色来完成示例。你也应该尝试不同的组合然后制定自己的一套方案（另见彩插）

3.8 小结

在这章里面，你学到了关于色彩的原理、色彩与情绪以及建立网页配色方案等方面的知识。现在你已经选好了为项目准备的颜色，应该可以开始搭建样式页面了。但是在那之前，还需要补充一下关于排版和字体方面的概念，这样才能保证给老板们看的样式页是精益求精的。

第 4 章

字体和排版

你能找得到的专注于讲排版的书不计其数，这方面的知识既复杂又深奥，有些人穷其一生都在研究它们。我们可不用这样，虽然要建网站写程序，但是我们只需要知道关于排版、印刷的基础知识就好了。更重要的是，我们需要知道如何去应用这些知识。除了为 Foodbox 选择字体之外，我们还需要保证阅读网站过程的流畅性，并保证网站本身的可读性。

排版不仅仅是字体的选择，排版的真正意义在于使你的内容具有可读性。文字是应用用户界面的核心，所以你对用户界面的要求会直接影响到对于字体样式、大小和空间等方面的选择。传统意义上，排版师的工作是将关于排版的理论运用在设计当中，让阅读文字变得尽可能地简单。不管页面有多么漂亮，如果上面的文字连看都没法看，那作为设计师，你一定是失败的。

4.1 深入字体

如果你搞清楚了字体的几个基本概念，那么挑选一个好看可读的字体将变得很容易。成千上万的字体中，只有少数几个是合适的选择；其中有些适合做标题或招贴画，有些适用于大段文本。

所有的字母都是基于一条基准线的（参见图 4-1），而小写字母 x 的高则被用来定位等分线。等分线和基准线之间的高度被称作一种字体的 x 高度。

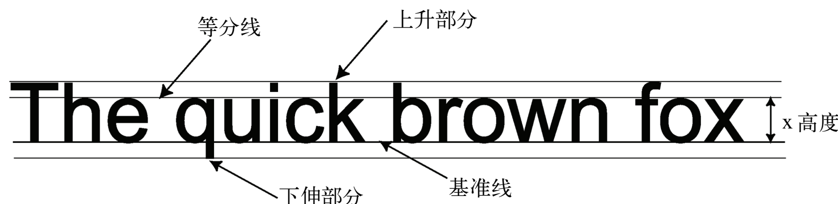


图 4-1　字体的组成部分

如果相对大写的 X 来说，一种字体中的小写 x 显得很高，那么通常会说这种字体拥有比较大的 x 高度。很多设计师都认为 x 高度比较大的字体更容易阅读，因为有些字母分辨起来会更简单。但是也要注意，如果字体的 x 高度过大，单词的可读性将变得极差，因为这种字体组成的文字看起就跟全大写的文本一样。正常的句子（It's much easier to read a sentence composed of mixed-case letters.）比全大写的句子（A SENTENCE COMPOSED ENTIRELY OF CAPITAL LETTERS.）读起来要容易得多。

有的小写字母（如 q 和 p）会有超过基准线向下延伸的部分；另外还有像 f 和 d 这种向上延伸高过 x 高度的小写字母。这些在基准线之下或者超过 x 高度的小写字母会影响可读性，因为这种字母可能会妨碍甚至遮住不跟它们在一条线上的字母。

4.2 字体类别

作为网页开发人员，主要需要关注三种字体：衬线字体、无衬线字体和等宽字体。这三种字体各有优劣，你要做的是在设计网页的时候考虑到这些优缺点。跟编程和设计所涉及的其他东西一样，字体也只是一种工具，重要的是要用合适的工具解决相应的问题。

4.2.1 衬线字体

衬线字体（serif font）很好认，这种字体中的字母通常会带有一条小尾巴——也叫衬线（参见图 4-2）。衬线字体中字母底部的笔画和末尾的笔画通常比中部的笔画要粗。微软的 IE 和 Word 软件的默认字体 Times New Roman 就是典型的衬线字体。然而这种字体是为印刷而设计的，在电脑屏幕上它的表现可不怎么样。

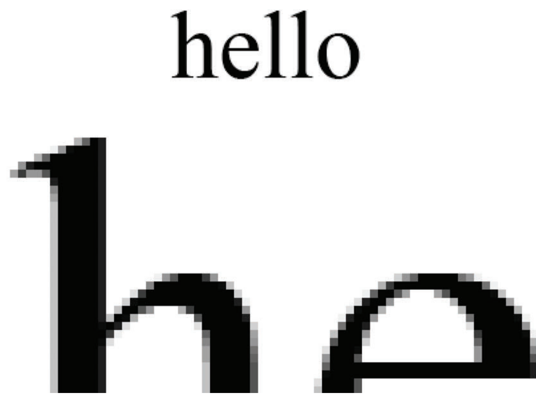


图 4-2 衬线字体的示例

衬线字体的主要问题是，那些太细的笔画使屏幕上显示的字母看不太清，如果你选用的字号比较小，这种情况会更严重。还应该注意的是在电脑上和在印刷中的情况是相反的，印刷的时候，衬线字体是一种可读性很高的字体。

在做标题、Logo 的时候，或者是需要很大文字的地方，衬线字体还是很不错的，它给人一种优雅、权威的感觉。

阅读障碍症患者可能更喜欢打印出来的衬线字体，这种字体中字母的独特性使其更好辨认。

4.2.2 无衬线字体

无衬线字体（sans-serif font）中的字母笔画从头到尾都是均匀的。顾名思义，这种字体就是“没有衬线”的字体（参见图 4-3）。Arial 和 Helvetica 是很著名的无衬线字体，Verdana 也是无衬线字体。



图 4-3 无衬线字体示例

无衬线字体适合在屏幕上阅读，所以它们是网站主要内容的首选字体。甚至在字号很小的时候，你也能很容易地阅读无衬线字体形成的内容。

4.2.3 等宽字体

在诸如 Courier 一类的等宽字体（fixed-width font 或 Monospaced font）中，每个字母所占的横向空间都是相同的。比如字母 i 和字母 w，虽然在衬线字体中它们的宽窄区别很大，但是在等宽字体中，它们所占用的横向空间是一样的。这种字体很适合显示源代码，或者是用来展现通过 E-mail 发送、内容只有文字的发票。

看看下面两张发票（参见图 4-4），其中一个采用了名为 Myriad Pro 的衬线字体，另一个用的则是名为 Courier New 的等宽字体，两张发票的内容都是一样的。可以看出用等宽字体的发票更好读一些，因为每个字母所占的宽度空间是一样，这样的排列看起来刚刚好。另外，采用等宽字体所形成的列看起来也很清爽。

```
Thank you for your order!

Item              Qty    Price
=========================================
Novelty Flying Disc    1    $5.00
Adhesive Bandages       2    $3.00
-----------------------------------------
 Subtotal:              $8.00
    Tax:             $0.00
 Shipping:              $5.00
=========================================
   Total:            $13.00
```

```
Thank you for your order!

Item                       Qty       Price
==========================================
Novelty Flying Disc         1        $5.00
Adhesive Bandages           2        $3.00
------------------------------------------
   Subtotal:                         $8.00
        Tax:                         $0.00
   Shipping:                         $5.00
==========================================
      Total:                        $13.00
```

图 4-4　使用 Myriad Pro（左）和 Courier New（右）的 E-mail 发票

4.3　字体限制及应对方法

Arial 和 Times New Roman 之类的标准字体遍地都是，所以很多设计师都喜欢在网页上运用独特的字体。然而 Web 上的最大问题是，并不是所有的电脑都支持所有的字体。Adobe Illustrator、Photoshop，甚至微软的 Word 软件都拥有大量的字体；你大可以在设计网页的时候随意使用这些字体，但是，很可能用户的电脑上根本就没有安装这种字体。

4.3.1　网页安全字体

实际情况是，真正的网页安全字体根本就不存在。微软列出过 5 种“微软网页安全字体”，这几种字体在大部分电脑上都可以正常显示（参见图 4-5）。如果你只用这几种字体，可以在很大程度上掌握内容的样式。即便是这样，你也无法保证所有的用户看到的内容都是一样的。

Arial
Courier New
Georgia
Times New Roman
Verdana

图 4-5　网页安全字体

这 5 种字体不是特别难看，但是它们并非原创，用多了也会无聊。事实上，它们已经被滥用了。很多网页都把 Verdana 和 Arial 当成标准字体，因为几乎所有的地方都有安装这两种字体。

最后，我们大概只能有这样的保证：每个系统上肯定都会有一种衬线字体、一种无衬线字体和一种等宽字体，而系统自己会为每种字体选一个默认的样式。另一方面，我们还有一些技巧可以突破这些限制，让自己能够控制的东西更多一些。

4.3.2 图片替换

设计师时常会把要显示成某种字体的一段文字做成图片，通常在制作公司 Logo 或者某个小节标题的时候会这样做。大多数设计师会在 Photoshop 或者 Illustrator 里做出样式，你也会碰到这种情况的。

在标题中使用图片化的字体不是个坏主意，但要注意不要滥用这个技巧，即便这个技巧能保证字体在每个浏览器中看到的都一样。因为将 PS 中的演示文件切割（slice）成网页[①]的过程会带来很多其他的问题。其中一个问题是页面下载的过程将耗费很长的时间——因为图片都很大。更重要的是，你的页面将无法被盲人使用，因为盲人依靠的是将文字转换为语音的屏幕阅读程序，而阅读程序是没有办法读出图片上的文字的[②]。我在 16.1 节讨论了更多这方面的内容。

设计 Foodbox 的时候，我们会用一种叫做*遮盖*的方法来解决这个问题。利用 CSS，我们可以用图片将大段的文字遮盖住，这样既可以保证在不同平台上的显示效果一致，又兼顾了可访问性。

4.3.3 用字体栈来定义备用字体

还有一种方法可以让你能够把握用户最终看到的字体样式：先列出你希望用到的特殊字体，跟在它后面的是用来替换的备用字体。如果用户的电脑上没有安装你最想让用户看到的字体，还有备用字体以防万一。最常用的定义字体的 CSS 代码如下：

```
body{
  font-family: Helvetica
}
```

上面的一小段代码定义了一种叫做 Helvetica 的无衬线字体，是 Mac OS X 上的常见字体，可是微软的 Windows 系统上并没有默认安装这种字体。于是，在 Windows 上的浏览器看到上面那

---

① Photoshop 中的切割功能能把图片变成 html 代码，又称“切片”。——译者注

② 虽然 `alt` 属性可以部分解决这个问题，但是面对大片的文字，`alt` 还是无能为力，所以我们不建议用图片代替大片的文本。

段代码后，就会尝试着去找这个并不存在的字体，结果没有找到，它会使用默认字体（Times New Roman）来代替，而这是一种衬线字体。

两者之间的差异是巨大的，除了有无衬线这一点之外，这两个字体在相同字号下的大小也是不一样的，Helvetica 字体的宽度和高度都略微的大一点。为了解决这个问题，你可以定义一种备用字体，当浏览器没有找到首选字体的时候，还可以采用备用字体。另外，你还可以定义多种备用字体，因此常常可以看到这样的样式定义：

```
body{
  font-family: Helvetica, Arial, sans-serif
}
```

这一段字体会告诉浏览器先找找 Helvetica，如果没有就试试 Arial，如果还没有，就用系统默认的无衬线字体。这并不是完美的解决方案，但是大多数情况下效果都还行。很多人把这种方法称作字体栈。

4.3.4 选择备用字体

单单知道如何使用字体栈是不够的，更重要的是学会组织字体栈中的结构。备用字体要和首选字体很接近。例如 Arial 和 Verdana 都是无衬线字体，但是 Verdana 有点太宽了，比较而言 Geneva 就更适合做 Arial 的备用字体。

在准备字体栈的时候，记得有个从特殊到一般的过度，先选择你的首选字体，再找一种让你看着舒服且跟首选字体相近的字体，注意两种字体的高度、宽度、x 高度、向下和向上超过基准线的高度都差不多，这样才能保证在替换的时候，版式不会受到太大影响。然后再挑一种你喜欢的网页安全字体，注意宽度要跟上面两种差不多。最后要用 CSS 指明一种浏览器可以提供默认的字体类（font-family），具体的值有：serif，sans-serif，monospaced，cursive 和 fatansy。浏览器会根据这些值提供默认的字体效果，CSS 代码如下：

```
p{font-family: Trebuchet, Lucida Sans, Arial, sans-serif;}
h1{font-family:Verdana,Geneva,sans-serif;}
h2{font-family: Baskerville, Times New Roman, Times, serif }
```

Unit Interactive 上有一篇很好的关于字体栈的博文[①]，还提供了一些不错的例子。

4.4 挑选字体

如果想为页面挑选一种有效的字体，先得考虑一下页面的内容。网页上会有菜谱，所以要保

① http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/。

证我们的字体容易阅读，不会让用户觉得困惑。同时，如果整个网页都用一种字体，那看起来也会不太对劲，所以在导航菜单、小节、页面标题以及另外的区域中，我们可能会选用不同的字体。当然，我们也不希望页面成了一个字体展示板，因此最好的方法可能是把每一页的字体上限控制在两个，当然，Logo 的字体不包括在内。我们会给网页的主体内容用一种字体，标题用另一种字体。

4.4.1　页面内容字体

大多数设计师都认为无衬线字体更适合内容展示。虽然有个别字母有点难以辨认，但是完整的单词在大多数显示器上都能被完美显示。

很多网页选用 Arial 字体，用 Helvetica 做备用字体。也有些设计师喜欢用 Verdana，这种字体比较宽，可以比 Arial 更好地填充空间。但是在字号很小的地方要避免使用 Verdana——在字号小于 10 像素的时候，它会变得难以辨认。[①]

4.4.2　标题字体

作为一个标题，你希望用那种可以抓住人眼球的字体，通常情况下还会使用更大的字号。有设计师习惯用内容字体的粗体来做标题，也有人喜欢采用一种完全不同的字体。后者可以让内容很多的页面看起来更加生动。

在为标题选择字体的时候一定要小心，要确保用户能容易地辨认它。优雅好看的字体不难找，但是要考虑这个问题：这种字体是不是能够帮助用户简单有效地区分页面上的不同部分。

正如你在图 4-6 中所看到的，我会选用 Monotype Corsiva 字体，这种字体很适合我们的标题。虽然它不是标准字体，但是作为标题，我们可以用把文字变成图片放在页面上。等会儿我会详细说明方法。

The quick brown fox

图 4-6　Monotype Corsiva，一种优雅的字体，适合做标题

① 千万，千万不要使用 10 像素以下的字号，不管你用的什么字体。没有任何借口去使用那么小的字，读这种字简直就是种折磨。

@font-face

在不远的将来，你就可以用@font-face来为页面链接自定义字体了。只是这个特性在老旧的浏览器中支持得不是很好。Firefox 3.5和Safari 4支持@font-face，但是它们之前的版本并没有提供这个特性。IE倒是从第6版就支持@font-face了，但是你必须用IE自家的受版权保护的字体。[①]

很快，你就可以这样定义字体了：

```
@font-face {
  font-family: "YourFont";
  src: url(/fonts/yourfont.ttf) format("truetype");
}
h1 { font-family: "YourFont", sans-serif }
```

这个方法既简单又方便，只是有个小问题：大部分字体（比如，photographs）都需要购买使用许可，才能用在这里。它们是有版权的，而你也得尊重版权。用@font-face不像用图片或者Flash——你在真刀真枪地传播字体，因为用户的浏览器会下载你所使用的字体。

幸运的是，类似于Typekit[②]的服务正在和字体作者还有发行商合作，试图解决这个授权问题。像Typekit自己有字体服务器来提供字体下载，开发者只需要在页面上加载一小段JavaScript记录Typekit账号就好了。所以，虽然现在就开始强推这个技术似乎显得有些急躁，但一定要关注它，毕竟@font-face大大简化了一些过程。

4.5 使用基线网格

为了将内容有效地传达给读者，文本的“流动性”是必不可少的。文字应该分布在图片和其他元素的四周，文字栏应该排成一行，并且栏中的每一行的换行位置不能让人觉得奇怪。大多数网络开发新手都会用浏览器的默认设置来安排文字流（text flow），但是如果你能在动手之前花些时间弄明白这些概念，设计出来的东西会显得更整洁。

基线网格是一种纵向排列的网格，也就是一些纵向排列的横线，这些横线支撑着文本中的文字。横线之间的距离可以作为基本计量单位（units of measure），横线本身则是所选字体的基准线。

基线网格中的横线的作用就像是纸质笔记本里的横格子，用来约束文字，使页面上每行的间隔均匀分布。为了让文字能够流动分布在文字栏和图片周围，需要把图片或者其他东西都跟基准线对齐，图片的高度应该是网格中两条基准线之间距离的整倍数。当所有东西都同基准线保持对

① 实际上Safari从3.1就开始支持@font-face了，而有消息称IE4就已经支持它了，详见http://www.evotech.net/blog/2010/01/font-face-browser-support-tutorial/。——译者注

② http://typekit.com。

齐之后，文字栏中的文字将自动分布在图片周围，并且间距均匀。最终，页面上的东西阅读起来将会相当轻松。

对比图 4-7 和图 4-8，你能感受到在用了基线网格之后，图 4-8 中的布局变得多么优雅。

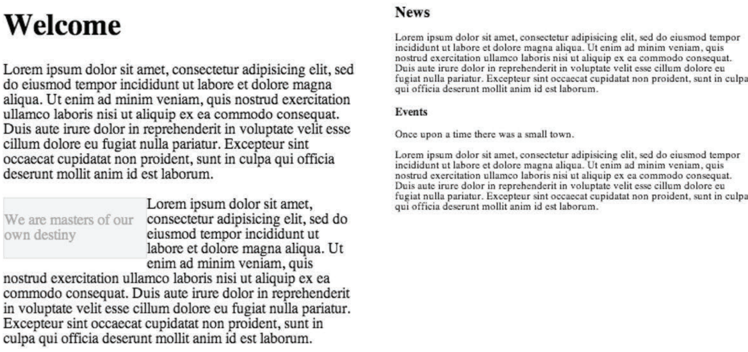


图 4-7 在没有横间隔的时候，两栏的文字并没有对齐

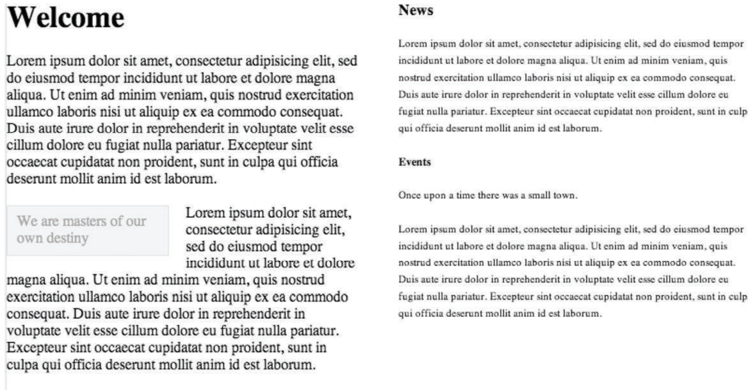


图 4-8 用基线网格可以让两栏中的文字行对齐

4.5.1 行距

行距（leading）指的是个两条基准线之间的距离，通常也称作 line spacing，在 CCS 中这个属性叫做 line-height。行与行之间的空白方便人眼浏览每行文字，同时它也是建立网格的关键。如果确定了行距，那么行与行之间的空白距离也就随之确定了。页面上的所有东西应该是这个距离的整倍数，这样才能保证这些元素都分布在基准线上。

4.5.2 计量单位

网格是基于文字行高的——就是两条基准线之间的距离。如果行高是 18 像素，那么文字就

得和基准线距离为 18 像素的网格对齐。

我们用像素（pixel）来定义基线网格中的基础字体大小，这意味着我们使用的是绝对值测量法。有些网络开发者觉得如果用了像素作为单位，用户将无法自己调整文字大小。其实这种说法不完全正确：老版本的浏览器确实不支持缩放像素单位的字体，但是大多数现代浏览器都支持这个技术了，甚至还能缩放行高。

当然，我们可以花些时间来挑选合适的字体，并根据这些字体来计算出行高、间距和其他的东西——然后面对不同的浏览器的时候我们会发现自己做的是无用功，因不同的浏览器处理这些东西的方法是不一样的。一些简单的搜索可以让你找到一些方法来把字号做成相对大小，但即便你找到了看似完美的方法，页面上图片的宽和高还是用像素来衡量的，于是你又不得不去找窍门来对图像做相应的缩放。

相对字体曾被看做是提高有视觉障碍用户的可访问性的救星，因为用户可以在浏览器中增加字体大小。但是这样做往往让事情变得更糟糕，因为图片不会随着文字而变大变小，后果就是文字流变得很奇怪，可读性降低。幸好，现在我们有更好的解决方法。

不管原作者用的是什么计量单位来划定的字体，微软公司的 Word 程序和 Adobe 公司的 Acrobat 程序都允许用户在自由缩放的同时保留原有的布局样式。这种方式已经被主流浏览器所支持。在 16.2 节，我会更加详细地谈到网页的可访问性和为有视力障碍的用户调整网页。



小乔爱问……

我看的所有书都说要给用户缩放字体的自由，你现在说的话是好建议吗

在第 1 章，我曾提到我是先天性白内障患者，视力非常差。我跟网络上的“小字体”们打过好长一段时间的交道了，有不少好心的开发者从所谓的“可访问性专家”那里得到了号称“最好”的解决方案，便如获至宝。可惜他们总是忘记测试这些方案，这种网页我也用过不少。实际上，视力差的用户会用类似 ZoomText 这种全屏缩放的辅助设备或者是 Windows 7、Mac OS X 自带的类似的东西。当用户读 Word 文档的时候，他们不会去调整文档的字体大小，而是用 Word 的缩放工具。用户用浏览器的时候也会这样做。而且，所有的主流浏览器都支持全页面缩放了。

回到 2001 年，开发者为了低视力的用户而饱受折磨地调整页面似乎还情有可原。现如今，浏览器已经跟上步伐，再为了图片能够随文字缩放而去钻研 CSS 和 JavaScript，就有点做无用功的感觉了。

4.5.3　为Foodbox选择字体

首先，需要一个基准字体大小，这样才能开始建立网格。我们会使用 12 像素作为主体文字的字号。这个字号大小适中，在一般的显示器上读起来也不费劲。[①]同时，还要给文字的上下都留下一定的空间，避免文字纠缠在一起而影响阅读体验。当然，空白留的太多也不好。同时，还要注意行高可以被整除。所以我们就用 18 像素的行高吧，这个高度能够在文字上下留出合适的缓冲区。高手的诀窍是将基准字体的大小乘以 1.25 或 1.5 得到的值作为行高。12 像素的字体就用 18 像素的行高，可以用 12/18 来表示。这种表达经常被印刷工人用来表达字号和行高。

现在，我们确定了 12 像素的主体文字字号和 18 像素的行高，为了保证网格发挥正常的作用，页面上的任何元素都要采用 18 像素的高度。这意味着页面顶部或底部的留白需要是 18 像素的倍数（或者加起来是 18 像素，比如 9 像素和 9 像素）。当你需要为页面增加纵向空间的时候，确保它是 18 的倍数，这样才能保证页面上的元素都能和网格对齐。同样，图片的裁剪也需要注意图片的高度是 18 像素的倍数，或者用 CSS 的内间距（padding）来调整图像的高度使之符合这个标准。

二级标题的字号要比文字主体的大，我们把它增加到 18 像素，跟网格的高度相同。这样看起来应该不错，但是要记得在二级标题下面插入 18 像素的间距。

而对于大标题呢，我们会使用两倍于基准字号的大小——24 像素。这比网格中的行距 18 像素要大，所以我们需要把大标题的行距扩大到 36 像素来保证页面元素的对齐。

在考虑页面元素摆放位置的时候，我们会回过头来再继续介绍基线网格系统，包括留白、边框、内间距、图片高度等在内的元素都要遵守网格对齐，否则设计就会变得“形散”。

下面是关于字体的一个小结：

区域	字体	字号	行高
标题	MonotypeCorsiva	24像素	36像素
侧边栏标题	Monotype Corsiva	18像素	18像素
二级标题	Arial	14像素	18像素
主体	Arial	12像素	18像素

你也可以随意组合这些设置来试一试，不要亦步亦趋地跟着我做，要有创意！譬如你可以试着调整字体大小来看看字号对页面样式的影响。

---

① 如果你的主要用户大多使用 24 寸 iMac 和极高的分辨率，这个字体显然太小了。当然，遇到这种情况，话题又会回到“了解你的用户”。

4.6 小结

排版在网页设计中是个重要的环节。如果没有考虑到字体的使用对阅读的影响，那设计出来的页面很可能让用户难以从页面内容中抓住信息。定义一个网格系统来标准化布局可以提高可读性，也会让页面更加美观。

选好字号和样式之后，我们可以往前再进一步，来做数字版本的样式页了。下面一个任务是为 Foodbox 设计 Logo。

Part 2

第二部分

图像设计

本部分内容

第 5 章

为 Foodbox 设计 Logo

我们的原始设计草图包含 Foodbox 的 Logo。你时常会碰到要在已有的 Logo 的基础上修改或者再设计的情况，客户也有可能把 Logo 交给第三方来设计，你则需要将完成的 Logo 融合在页面里。就 Foodbox 来讲，我们需要根据现有的网页设计来完成一个新的 Logo，因为上一版的网站并没有提供一个 Logo，也没有现成的数字化的网站可以参考。你可以参见图 2-4 来设计 Logo。

5.1 建立工作目录

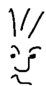
将工程组织得有条有理，你将受益无穷。如果开发过 Ruby on Rails，你会明白采用标准化目录结构所带来的好处。虽然在做设计的时候没有这种类似的标准，但是大多数网页设计师都有各自的办法来组织记录工作轨迹。在目前这个项目中，我们也会采用一种简单的目录结构来维护样式和图像文件。

新建一个叫做 Foodbox 的文件夹，在其中再建立三个子文件夹，分别叫做 images（图片）、stylesheets（样式表）和 originals（原始文件）。[①]

originals 文件夹会用来放置所有的工程文件——比如 Illustrator 或者 Photoshop 的文档，以及客户给你的所有图像。images 文件夹则用来存放网页需要直接使用的所有图像。而 stylesheets 文件夹则是用来存放页面的 CSS 文件的。

---

① 文件夹最好用英文名以防止路径问题发生。——译者注



小乔爱问……

我必须要用 Illustrator 吗

也不是，但需要指出的是，如果想要从事设计相关的工作，Illustrator 是一个很好的工具，很有学一学的必要。所以学会使用 Illustrator 有利无害。Adobe 的网站上可以下载 30 天的试用版，足够你掌握这一章的内容了。如果实在不想用 Illustrator，Inkspace[①]也是个不错的选择。本章内容是基于 Illustrator 的，如果你用的是其他矢量图形工具，你可能需要把这里讲的操作转化为该工具里对应的操作。

5.2 Foodbox 的 Logo

用矢量图形来设计 Logo 是一个很重要的原则，只有这样 Logo 才可以被缩放到任何合适的大小，然后就可以把 Logo 放在网页甚至印刷品上。业界以 Adobe Illustrator 为标准，我们也会用这个软件来重做一个 Foodbox 的 Logo。

新 Logo 由四个方形和文字“foodbox”组成。完成作品参见图 5-1。只需要简单的几步，你也能做出一个那样的 Logo。



图 5-1 完成了的 Foodbox 的 Logo

① http://www.inkscape.org/。

打开 Illustrator，新建一个文档，文档大小随意，因为将 Logo 导入 Photoshop 文档时，我们会进行相应的缩放工作。现在，只是用 Illustrator 做个 Logo，所以我不会讲得太详细。但是如果你准备做更多的图形方面的工作，我建议好好研究一下 Illustrator，这是个很棒的工具。

先从四个方形开始吧。我们需要四个圆角方形排成两排，每排两个。我们只需要很少的步骤就能画出一个特定大小的方形，然后用 Illustrator 的复制功能复制出剩下的几个。

(1) 在工具板里选择 Rounded Rectangle 工具（圆角长方形工具）——按住 Rectangle（长方形）工具不放，就会有其他的方形绘制工具出现，其中就有 Rounded Rectangle 工具，选中它。

(2) 在 Options Toolbar（选项工具栏）中，将填充色设置为#FCEE21，边框色设为黑色。

(3) 双击画布，会出现一个对话框，让你输入方形的边长。设置方形的长和宽都是 100pt[①]，角的弧度是 12pt，单击确认，即可得到一个固定大小的方形。

(4) 然后，双击工具板中的 Selection（选择）工具，会弹出 Move or Copy（移动或复制）对话框。

(5) 我们的目的是复制这个方形，并且让复制得到的两个图形之间有准确相等的间隔。在 horizontal（水平）一栏中输入 110pt，然后单击复制按钮。这个操作会在距离刚刚那个方形的起始点 110pt 的地方复制一个方形。这样，两个方形的距离就是 10pt 了。



小乔爱问……

等等！Logo 中的不同颜色是怎么来的

我用到了调色板中的不同颜色，调整了方形的色调，让它们看起来更好看，同时我还打印了几份出来，为了确保打印出来的和屏幕上的看起来差不多。

(6) 用相同的方法复制出剩下的两个方形，只不过这次会用到垂直距离而不是水平距离。用选择工具选中刚刚得到的两个正方形，然后双击工具板上的 Selection 工具打开 Move or Copy 对话框。在垂直距离那一栏中填入 110，水平距离设为 0，单击复制按钮，现在你得到了平均分布的四个正方形。

接下来，要为这四个方形上色了。

(1) 按下 F6 打开调色板（Color palette）。

(2) 用 Direct Selection 工具直接选择工具选择一个方形，双击调色板中的填充色方块。

① pt 是 Illustrator 中的一个单位缩写，代表的是“点”（point）。——译者注

(3) 从左上开始，按照逆时针顺序分别给四个方形上色：黄（#FCEE21）、绿（#C2EE21）、橙（#FCBA21）和米黄（#FCEEB5）。

现在是时候往里添加文字了。为了让 Logo 看起来平衡一些，需保证“foodbox”几个字的高度和两个方形的高度相同。为了达到这个目的，需要用到*定位辅助*（guide）。

大多数的绘制工具都提供了定位辅助的功能，用来帮助对齐对象，或者帮你将绘制的图形安置在它应该在的位置。对于那些有过设计工作经验的人来说，定位辅助的概念并不会显得陌生。我们将会使用这个功能来定位文字，方便又简单。

(1) 确定标尺没有被隐藏，按下 Crtl＋R。

(2) 在方形的最上沿设置一个辅助线。在图像的顶端有一个水平的标尺，将鼠标放上去，按住鼠标并向下拖动鼠标，就可以得到一根辅助线，将辅助线放置到方形的上沿，放开鼠标，这样上沿的辅助线就设置好了。

(3) 如法炮制，得到方形下沿的辅助线，这样就可以根据辅助线来加入文字了。

现在，你应该看到图 5-2 中的两条辅助线了。

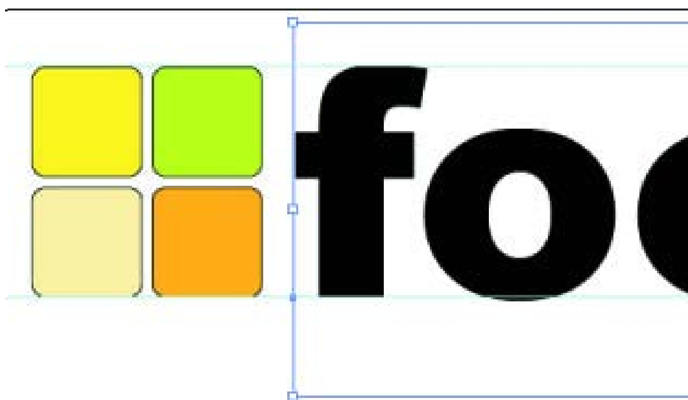


图 5-2　缩放文字

然后开始添加文字。

- 选用 Text（文字）工具。
- 在文字选项板中设定字体为 Arial Black，字号为 72pt。
- 在画布上输入“foodbox”，随便把字放在那里都可以，我们一会儿再移动它们的。
- 接着单击选择工具，输入的字的周围会出现供你调整大小的手柄。按住 Shift 键的同时拉动右上角的调整手柄，让 f 的上端接触到靠上的那个辅助线，同时也要保证 f 的下端接触

到靠下的那根辅助线。如果效果不理想，多试几次缩放和移动直到满意为止。如果其他的字母超过了靠下的辅助线，不要担心，我们马上就会处理。

创建字符轮廓

设计师经常用 Illustrator 中的创建轮廓（Create Outlines）来调整应用在文字上的字体。但是这个功能还有其他的好处。

我接到一个客户的时候，会问他有没有现成的 Logo。如果有，我会设法要来一份 Illustrator 或者 EPS 格式的文档，这样我就能调整 Logo 让它可以适当地放在网页上了。偶尔，我拿到的 Logo 中含有特定的字体，我不得不去寻找免费版本或者花不少钱去买一份授权（这种情况更常见）。

有种折中方案是找到 Logo 设计师做一份 Logo 的副本，拿到副本后我会用创建轮廓工具来得到图案中文字的轮廓，然后利用这些轮廓来设计。这样既保留了字体的样式，又兼顾了跨操作系统和跨平台性。

仔细观察，会发现有些字母的下沿超过了辅助线。可以微调字体来解决这个问题。

(1) 用 Select 工具选择字图图层。

(2) Select Text>Create Outlines。这个命令将文字转化成矢量图形。现在不能改动文字内容了，但是可以用任何绘图或者操作工具来调整文字的形状。

(3) 用工具板中的 Direct Select 工具，框选字母 o 的下半部分，按几次向上按钮，直到 o 的下沿接触到辅助线。

(4) 对其他超过辅助线的字母重复上面那个步骤。

最后，单击选择工具，按下 Ctrl＋A 来选择所有对象。按住 Shift 键的同时调整 Logo 的大小让它能够放置在画布的边界框之内——就是那个画布上用黑色边线框住的长方形中。

将这个文档保存成名为 foodbox_Logo.ai 的文件，放置在项目的 originals 文件夹中。稍后我们会将这个文件导入到 Photoshop 中，所以当你保存文件的时候，记住要勾选 PDF 兼容选项。否则 Photoshop 无法导入这个文件。

记住我们用的是矢量绘制工具，所以从咖啡杯到大广告牌，我们都可以用这个文件。Logo 可以被无损地任意放大或者缩小。

5.3 当我们需要自己设计 Logo 的时候怎么办

在上面那个例子中，我们有一个现成的参考，所以我们其实是在学习 Illustrator 的基本操作，

然后对原有的东西进行再创造。那么，要为你的产品和生意从头开始设计一个 Logo 的话，还需要其他什么东西呢？

想想现如今的一些成功的 Logo 吧。全世界的人都认得可口可乐的 Logo，耐克的也很好辨认。但是，这两个 Logo 跟它们所代表的产品一点都不像。

其实，设计 Logo 和设计网页的方法差不多。

相对字句，人们阅读图像的速度要快得多，设计 Logo 就要充分利用这一点，让人一看到 Logo 就能联想到公司的产品。比如说可口可乐的 Logo[①]，人们往往不用阅读其中的文字，就能将这个 Logo 和它的产品联系起来。这才是设计 Logo 的最根本目的——你的 Logo 只代表你和你的公司。

人们对 Logo 的快速辨认基于公司对 Logo 的长期和稳定的使用。如果你时常更换 Logo，人们就很难建立起你所需要的品牌辨识能力。Logo 就代表了你，人们会牢牢记住它，如果 Logo 设计得不好，它也就起不到相应的作用了。律师事务所的 Logo 与水上乐园的 Logo 一定是迥异的。

如果你设计的 Logo 包含了文字，那么一定要让文字可以被很容易地阅读。记得使用清晰易读的字体，要么把字体弄得很大，要么就用很小的字号。配色方面则要简单，而且要用比较保险的颜色。回忆你学过的颜色和情绪的关系，把它运用到 Logo 设计中。

跟网页不同的是，Logo 很可能被打印出来，所以要用打印机把 Logo 打出来，看看颜色是否正确。当进行网页相关的制作时，用 RGB 色彩模式，但是如果你设计的东西会被印出来，CMYK 应该是你的选择。基于 CMYK 的图片可以被转存为 RGB 模式用在网页上，但是逆操作则非常困难，特别是颜色匹配方面的问题更令人头疼。

最后，别忘了在黑白模式下检查 Logo，你可以打印一份黑白的 Logo 看看效果。

5.4 小结

像 Illustrator 这种基于矢量的图像很适合创作可缩放、多功能的 Logo。下次你要设计 Logo 的时候，别忘了用上这章所学到的技巧，比如复制和字体修改等。更别忘记要有创意，你可以随意修改上面那个设计好的 Logo，做出一些改变，用不同的形状、字体、字号或者布局，甚至可以根据上面的建议设计创造一个崭新的 Foodbox 的 Logo。

下一步的任务是，做出一个彩色的样式页。

① 可口可乐的 Logo 是 Coca-Cola 这几个字组成的。——译者注

第 6 章

设计样式页：页面结构

前面几章我们已经完成了设计草图和配色方案，接下来就可以开始使用 Adobe Photoshop 制作 Foodbox 的首页样式了。这一章会拟出页面的结构，并且把页头（header）和页脚（footer）做好。慢慢地你就会熟悉 Photoshop 当中的一些布局选项，这些选项可以帮助你在之前定义的网格中对齐页面中的各种元素。

6.1 关于图层

图层非常强大，除此之外我想不出还有什么形容词可以用来描述它的作用。有了图层，你可以把一个整体的作品打散成为不同的模块，便于创作和管理。每一个图层就像是一个独立的文档。你可以对单独的图层进行粘贴、复制、选择、删除，甚至在图层上应用各种效果。同时，图层还是透明的，这意味着你可以将它们叠加起来形成完整的作品。图 6-1 演示了样式页里组合图层的方式。

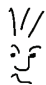


图 6-1 图层组合

很多时候，设计师会在一张图片上新加一个图层，在上面写些文字。在这种情况下，文字不是直接写在图片上的，所以设计师可以随时修改它们，同时保留了 Photoshop 文档中图片的原貌。

当你从 Illustrator 或者 Photoshop 中导出 JPEG、GIF、PNG 或其他格式的图片时，软件会自动合并图层。但是如果你把原始文档弄丢了或者删除了，那就不得不从头开始了，文档中的图层是不能从图片中恢复的。这也是导致很多 Logo、按钮或者其他图片的制作总是需要从头再来的原因。

下面，我们会在 Illustrator 和 Photoshop 中用到大量的图层，你要注意的是随时保存原始工程文件。



小乔爱问……

我必须要用 Photoshop 吗

不一定，但是推荐用它。理由跟我在第 5 章的“我必须要用 Illustrator 吗”中说到的一样。Photoshop 是处理照片和光栅图形[1]的行业标准。虽然从技术上讲，用其他便宜一点或者开源软件也能完成我们的任务，但是在本书里我还是以 Photoshop 为例。

当然，你也没必要现在就冲到商店里买一套 Adobe Creative Suite，只是为了这本书就这么干不太值。Adobe 提供了 30 天的试用版，足够让你学会书中的例子了，然后你可以决定是不是真地需要买一套软件。只要你在 Photoshop 中完成了这些练习，你会发现在别的软件中的操作也是类似的。

当然，你也不必非得用最新版本的 Photoshop。书里的例子应该可以在任何版本的 Photoshop 中完成。

如果你还是坚持己见，那我向你推荐 GIMP[2]或是它的修改版 Gimp-Shop[3]。这个修改版做过优化，看起来更像 Photoshop 一些。你可以用这些软件完成本书的例子。

6.2 基本结构

最开始，我们需要把主页的设计草图变成由长方形组成的一个个区域：页头、页脚以及两个内容栏，把这些区域规整成长方形（参见图 6-2）。这一步的关键是用长方形的思维来思考页面结构，实际上，在平时看网站的过程中，你就应该用这种思维来寻找页面上的长方形。

① 点阵图像或位图图像，由像素点组成。——译者注

② http://www.gimp.org/downloads。

③ http://www.gimpshop.com/download.shtml。

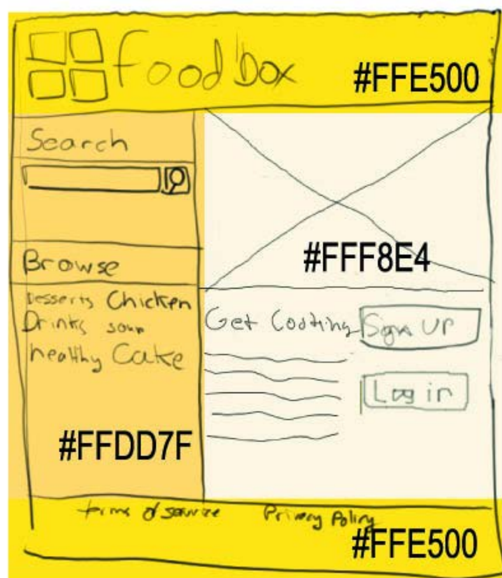


图 6-2 用颜色标记的不同区域

6.2.1 屏幕大小

制作网页的时候，我们不可能知道用户会使用多大的屏幕观看网页。最好的方法就是先以平均屏幕大小为目标。譬如说在写这本书的时候，网页用户的屏幕大多数都是 1024 像素×768 像素的分辨率，还有很大一部分的屏幕分辨率更高。[①]

但是这个数据可能会有一些误导性。就算用户用的是宽屏的显示器，他们也不一定会用全屏方式来使用浏览器，而是将浏览器窗口跟别的程序并排放在屏幕上。同时，人们还会用手机、PDA、iPod 甚至任天堂 Wii 游戏机来上网。你的网页在任何大小的屏幕上都要有可读性。

在 Photoshop 中新建一个名叫 foodbox_mockup 的文档，将它的宽度设为 900 像素，高度设置为 756 像素，分辨率为 72dpi（点/英寸）。色彩空间设置为 RGB，背景色是白色。之所以这样设定，是因为我们把 1024 像素×768 像素的分辨率作为目标屏幕。这样一来页面的宽度就比屏幕的宽度要小，用户不用滚动横向的滚动条，同时页面的两边还会有一定的留白。我们将会制作一个定宽的页面。

新建好文档之后，将文件存在 originals 文件夹中，文件名设定为 foodbox_mockup.psd。

---

① 根据 http://www.w3schools.com/browsers/browsers_display.asp 的数据显示，写这本书的时候，54%的用户的显示器分辨率是 1024 像素×728 像素，另外有 26%的用户的分辨率要大于这个值。

6.2.2 定宽布局

在定宽的布局中，网页的大小是固定的，不随浏览器窗口大小的改变而改变。这种布局比起可伸缩的（或者叫液态的）布局更容易设计和实现。液态布局需要更多的测试和代码才能保证其在不同浏览器中的可读性。只要稍有疏忽，没有对液态布局的代码做恰当的维护，页面中的文字行就可能变得极长而溢出页面，或者是太短而不能充满页面。无论上述哪种情况发生，页面都会变得极难阅读。相反，实现一个静态布局则不需要花费多少时间。

然而，不同的页面需要不同的布局。那种提供大量信息的网站可能需要流动的页面；展示大量数据的商用网页程序可能并不适合定宽的页面。你需要根据情况量体裁衣，定制不同的页面布局，而不是随波逐流，也不要在每个设计中重复使用单一的模板。

折叠

我所使用的这种页面长宽可能会导致网页的一部分被“折叠”起来——意思是小显示器前的用户不得不往下拉动滚动条才能看到完整的页面。“折叠”一词来自于印刷界，指的是在制作报纸的时候，会尽量在第一版的上半部分塞入更多的信息，这样人们就能在不展开折叠的情况下阅读到更多的内容。然而，这个理论现在不再适用了。虽然人们不太愿意向下拖动滚动条，但是大家都已经习惯了。所以只要你设计的页面能够清楚地显示出有向下拖动滚动条的必要，用户们通常会乐意这么做。

6.2.3 设置网格

在 4.5 节里讨论排版基础的时候，我们曾经深入地讨论过基于网格的工作。同样，在 Photoshop 中也可以把网格调为可见，它会出现在画布的上方，我们可以利用这个网格来排列页面上的元素，比如文字之类的。但是 Photoshop 中默认的网格设置并不能满足我们的需求，所以需要作些细微的调整。

首先，选中 Edit（编辑）菜单下的 Preference（首选项）栏，在 Units and Rulers（单位和标尺）面板里把标尺的单位改成像素（px）。然后，在 Guides, Slices and Count（辅助线、切片和计数）面板中将 Guidlines Every（辅助线间隔）设定为 18 像素，这个值也是我们选的行高。同时将辅助线的颜色调整成为比较刺眼的颜色，像是橙绿色，或者是某种你在设计中绝对不会用到的颜色。这样，在工作的时候辅助线会显得更加突出。

在 Photoshop 默认的工作界面中，网格和标尺并不是开启的，按下 Ctrl+R 键可以唤起标尺，Ctrl+’则是网格的快捷键。

6.2.4 用辅助线划定区域

设计 Logo 的时候，你学会了如何用辅助线对齐 Logo 中的文字和四个方块。同样，你会发现用辅助线把构图分成几个不同的部分是很有用的，它可以帮助你绘制和对齐页面中的元素。

我们用 Rectangle Shape（矩形绘制）工具，从页头和页脚开始，照着草图画出页面上的不同区域。在这个练习里，你需要将页头和页脚做成同样的颜色，当然，别忘记做些实验性质的组合。

先从页头和页脚开始。首先，页头要足够高，这样才能放得下我们的 Logo；但是页脚就不用，因为里面只会显示版权信息和使用条款等。当然，在设定它们的高度的时候要记住之前讲过的基准线网格——我们定的网格高是 18 像素，所以页头和页脚的高度应该是 18 的倍数。试试把页头定为 108 像素，把页脚设为 54 像素。

要放置页头的辅助线，将鼠标指针移放到图片上方水平的标尺上，按住鼠标不放往下拖曳至 108 像素的位置。然后以同样的方法在 702 像素位置的页脚放置辅助线。

我们需要足够宽的侧边栏，用于显示菜谱搜索表单和标签云。所以把侧边栏设置为 306 像素宽。当然，之后还可以调整这个宽度。将鼠标指针移放置到图片左侧竖直的标尺上，按住鼠标不放向右拖曳至 306 像素的地方，松开鼠标，放下基准线。

现在，我们已经用基准线标出了页面上的四个区域（参见图 6-3）。然后用矩形工具向区域中填充形状。注意，四个区域的边界线都落在了之前我们讲的网格基准线上。现在开始打扮这个页面吧。

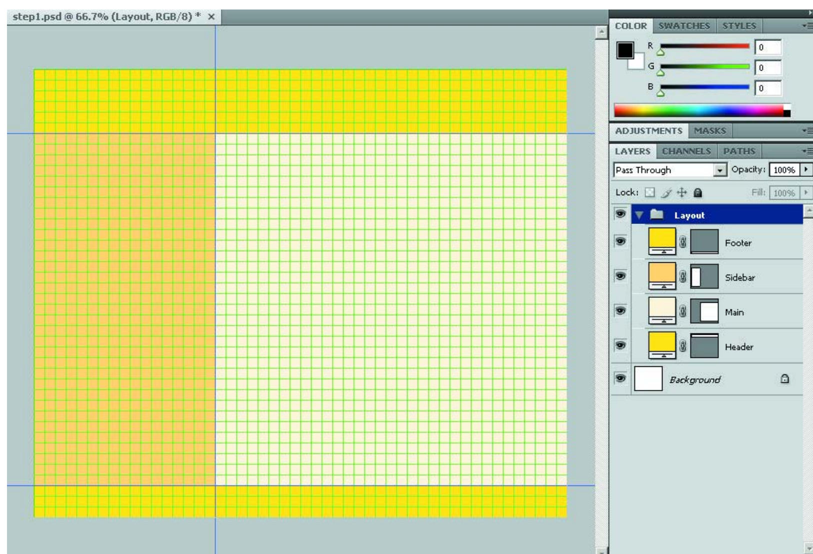


图 6-3 网页的四个区域

6.2.5 画出不同区域

接下来的工作是根据辅助线画出不同的区域。按下 U 键选择矩形工具，同时要注意勾选 Shape Layers（形状图形）选项。

在选项面板里，选中调色板，将颜色设置为 FFE500。单击 OK 按钮。在图像顶部画一个矩形，覆盖那条在 108 像素位置的辅助线上方的区域：现将鼠标放在画布的左上角，按住鼠标不放并拖曳至辅助线和画布右边框的交点。

然后是页脚。同时按下 Shift+Ctrl+N 键新建一个图层——把新的东西放在一个新的图层上是一个好习惯，这样改动起来也会比较容易。创建图层的时候 Photoshop 会要求你给图层命名，以便日后查找。从 702 像素位置的辅助线开始，画出页脚的矩形区域。

下面就轮到侧边栏了，为它新建一个图层，为这个图层取个恰当的名字，将颜色设置为 FFD67F，画出左边的侧边栏，侧边栏应该恰好在辅助线框起来的区域内，从（0，108）像素的位置按住鼠标不放，拖曳到（300，720）像素的位置。

然后将剩下的区域用颜色为 FFF7DF 的形状图形填满。

现在，四个框架区域都定义好了，接下来该考虑首页上其他元素的摆放了。草图中，Foodbox 的 Logo 是在页面上方的，而且 Logo 我们也已经做好了。保存一下文件，继续完成后面的工作。

6.3 放置 Logo

Photoshop 在网页制作方面很实用的原因之一是，这个软件与其他软件的兼容性很强。我们刚刚在 Illustrator 中画好了一个 Logo，现在，可以将那个画好的图片以矢量图形的形式导入到样式页中。然后可以对 Logo 进行缩放，将之放在合适的位置上。这时，辅助线就能帮住我们定位它的位置了。

先确定标尺和网格都是开启的，然后画两条水平的辅助线。第一条辅助线要画在 18 像素的地方，正好和网格中的第一条水平线重合，第二条辅助线则和网格中处于 90 像素的水平线重合。这样，Logo 的上方和下方各留出了 18 像素的空隙。然后从左边的标尺中拉出一条竖直的水平线，停在 18 像素的位置。这样，我们就确定好了放置 Logo 的准确位置。

单击 File 菜单下的 Place 选项，找到 Logo 文件，单击确认。将 Logo 拖曳至画面的左上角，让 Logo 的左上角跟刚刚画的那两条辅助线的交点重合。然后用缩放手柄来调整图形大小。按住 Shift 键，把 Logo 右下角的缩放手柄向左上方拉动，直到 Logo 的右下角碰到处于 90 像素的那条辅助线。按下 Enter 键放置文件。

当把一个 Illustrator 文档放置在 Photoshop 文件中时，这个文档会被当作智能对象（Smart Object）。如果你修改并保存了 Illustrator 文档，这些修改会自动反映到 Photoshop 文件当中（参见图 6-4）。

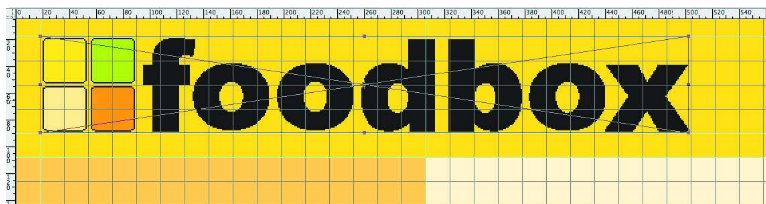


图 6-4 在 Photoshop 中的 Logo

6.4 用图层组组织图像

到现在，我们拥有了相当多的图层，要从中找到我们需要的东西就有点难了。好在 Photoshop 中有个功能可以让我们很方便地管理这些东西。所谓的图层组实际上就像一个包含了很多图层面板的文件夹，这样可以让操作变得方便。

创建一个图层组，需要在图层面板上单击图层组按钮。现在新建一个名叫 Layout 的图层组。对图层组重命名的方式如下：右键单击图层组，选择重命名选项。

然后把页头、侧边栏、主体和页脚都拖放到这个组里面。

图层组是可以收起的，收起之后，你就可以集中精力于当前的工作图层。图层组是可以开启、关闭甚至被复制的。这些功能可以让你轻易地把杂乱的东西理出头绪。接下来的几节中，我们会大量使用这个功能来保证文档是有序的。

6.5 给 Logo 加上倒影

Steve 想起来有位领导说想给 Logo 加上倒影。现在有很多页面都采用了这个技术，最终效果就像把文字或者 Logo 放置在一个湿的地板上，这样文字或者 Logo 就会像站在它们的倒影之上。用图层组和蒙版的技术，这种效果实现起来很容易。

(1) 首先新建一个叫做 Logo 的图层组，将含有 Logo 的那个图层放入其中。在另外一个图层上制作 Logo 的倒影，我们希望这两个图层能组织在一起。

(2) 在图层面板里，右键单击含有 Logo 的那个图层，选择 Duplicate Layer（复制图层）。将复制得到的图层命名为 Foodbox Logo Reflection。

(3) 选中倒影图层，选定 Marquee（选择）工具，在图片上单击右键，选择 Free Transform（自由变换）选项，激活缩放手柄。单击图片上方中间的那个手柄不放，向下拖曳，越过图形底部，这样整个图片就被翻转过来，形成了倒影的效果。需要注意的是，倒过来的图片应该跟原来的 Logo 同高，你可以借助辅助线，也可以在拖曳的同时按住 Shift 键来帮助调整高度。[①]

然后按下 Enter 确认变形。用 Esc 键则可以撤销变形操作，从头再来。

(4) 倒影的淡出效果有很多种方法可以实现，最简单的方法是使用 Photoshop 的图层蒙版功能。选中刚刚经过变形的那个图层，单击图层面板下方的 Add Layer Mask 按钮。蒙版可以让一张图片的某些部分隐藏起来不显示，换句话说，图层被蒙版遮住的部分是不可见的。

(5) 如果在蒙版上用一个渐变图片而不是色块，那么蒙版下的图片或者图层就会有一个淡出的效果。从工具面板中选中 Gradient（渐变）工具——注意，有时候 Gradient 工具是不可见的，因为它跟 Paint Bucket（油漆桶）工具放在了一个地方。如果你看到了油漆桶工具，只需要点住它不动，在弹出来的菜单里面选择 Gradient 工具即可。

Gradient 工具的选项会出现在 Photoshop 窗口中靠上的位置。在选中渐变之后，你就应该能看到几个预制的渐变样式，选一个从黑到白的渐变样式。

(6) 设置好渐变工具，选中蒙版，按住 Shift 键的同时在图像上从上到下画一条竖线，从 72 像素的地方起笔，画到 108 像素的位置。

这样，渐变的黑色部分就形成了蒙板，从而得到我们想要看到的淡出效果。你可以多试几次以达到理想的效果（参见图 6-5）。

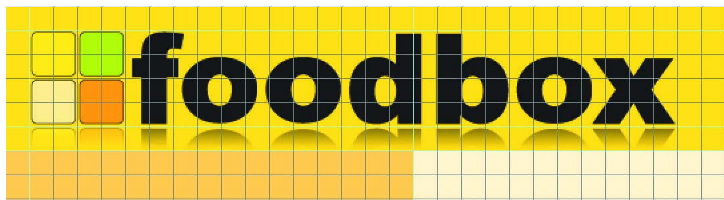


图 6-5 带有倒影效果的 Logo

6.6 页脚

跟上一版的网页一样，页脚中会有一些文字，包括版权信息、使用条款和隐私信息。

选用 Text（文字）工具，将字号调成 10 像素，字体设为 Arial。如果需要超链接，把超链接

---

① 如果你不想这样手工操作，也可以用 Flip Vertical transformation（竖直翻转变形）选项翻转图片，然后将得到的图片拖曳至原始图片下方。

文字设置成不同的颜色，并把版权信息、使用条款和隐私信息的链接文字放在不同的图层上。可以从配色方案中选一种颜色作为超链接的颜色。

没必要在这些文字上花费太多时间。不用追求完美居中之类的效果，因为这些东西迟早是要放到内容文档中的，现在只需要做出一个样式页，然后以此收集反馈。

6.7 小结

本章内容包含了一些关于 Photoshop 的内容，你学会了如何导入文件、图层相关事宜和用辅助线对齐元素。现在，开始往这个结构中填充内容吧。

第 7 章

设计样式页：内容相关

上一章完成了两个很重要的任务：首先是将页面划分成了四个区域，其次是设置好了页头和页脚。接下来的任务则是往侧边栏和主区域填充内容了。同时，你还要做出搜索框和标签云的样式来，这些东西将会放置在侧边栏中。在主区域里，我们会做一个 Banner 图片和一个文字版的推荐广告，并且会将这些元素放在合适的位置。

7.1 制作搜索框

在设计草图里，我们把搜索框放在了侧边栏的上部。搜索区域会有一个较大的标题，搜索框在标题的正下方（参见图 7-1）。接下来再一次利用基准线网格将元素放置在网格线上，而首先要做的则是新建一个叫做 search area 的图层组。这个图层组会包含本章内创建的所有东西。

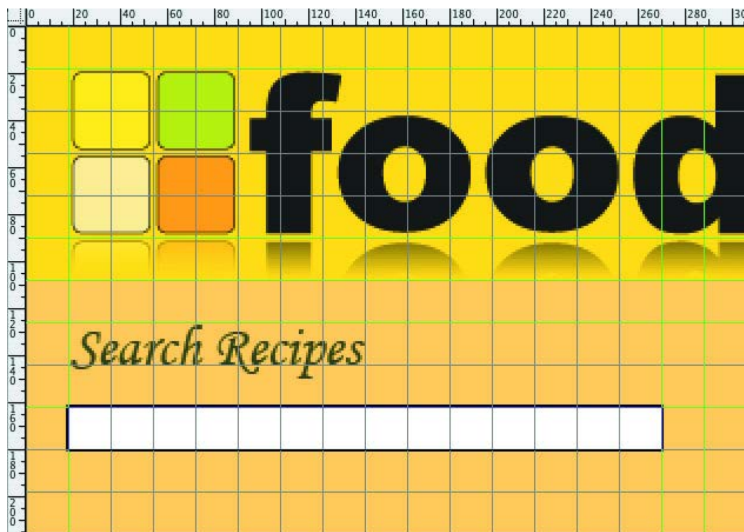


图 7-1　完成的搜索区域

照旧，辅助线将是放置元素的得力助手。先在左边标尺的126像素的地方放一条水平的辅助线，这条辅助线是搜索区域标题所在位置的参考。如果用标尺刻度放置辅助线的方式让你感觉到无聊了，你可以试试这种方法：输入朝向和位置坐标直接创建辅助线。具体做法是从View选项中选择新建辅助线，然后输入合适的参数创建。用来确定标题的第二条辅助线应该在162像素的位置。虽然由于标题的上沿会和126像素位置的辅助线重合，这个辅助线显得有点多余。然而，标题的字高有24像素，为了能让行高是18像素的倍数，我们需要额外的空间，这就导致标题的行高是36像素。

按下字母T键激活文字工具。将字体和字号分别设置为Monotype Corsiva和24pt，将文字颜色设为4B541C，这是在配色方案中我们为标题选定的颜色。在刚刚定义的那条辅助线的下方单击一下画布，输入“Search Recipes”。用移动工具让文字和辅助线对齐。

接下来要用绘制工具和一些图层效果来画出搜索框。首先需要用辅助线来划定搜索框的区域。现成的162像素位置的辅助线已经标记出了搜索框的顶部，我们还需要两条竖直的辅助线，分别位于270像素和288像素的位置，这样就确定了搜索框和按钮的位置。注意，每条辅助线都落在了网格上。

新建一个名为search box的图层。选用矩形工具，将填充颜色设置为FFFFFF，在辅助线所圈定的范围内划出一个搜索框。右键单击这个图层的小图标，选择Blending Options（混合选项）。从弹出来的对话框中进入Stock（描边）选项卡，将描边大小设为1，颜色为000000。

7.2 餐谱导航标签云

网站是使用标签的形式来组织菜谱分类的。用户可以为菜谱们打上标签，这样在大量的菜谱中可以方便地找到想要的东西。现在，不少受欢迎的网站都会用标签云（tag cloud）的形式来显示系统中最常见的标签。标签云会用不同大小的字体来显示关联次数较多的标签。例如，如果被打上“甜点”标签的条目是其他类型的三倍还多，那么“甜点”这个标签在标签云中的字号就要比其他的标签大很多倍。我们会做一个类似于图7-2的效果。通常来说，标签云需要5~6种不同的字体大小，但是作为一个样式页，大、中、小三种字体就够了。

还是先从标题开始吧。这里会用和搜索框标题一样的字体和颜色。在216像素和252像素的地方分别放置两条水平的辅助线，标题会被安放在其中。新建一个图层，按下字母T键选用文字工具，再确认一下文字颜色是4B541C，跟搜索框标题一样是那种带点绿的颜色。输入“Browse Recipes”，用Move（移动）工具将文字拖曳到两条辅助线的中间，然后将这些文字和搜索标题对齐。注意辅助线和标尺，要学会使用它们！如果用鼠标移动不是很方便，你可以尝试用上下左

右四个箭头来进行细微的调整。



图 7-2 完成的标签云

现在，可以开始用文字工具往页面上放标签了。举个例子，选择一个 18 像素的字号和 Arial 作为字体，选上加粗，把颜色调整为 54431C，键入“Desserts”并将之放置在辅助线之间。然后你可以分别键入几个不同的标签文字，每个标签自成一个文字块；同时要注意字体、字号的不同，要把它们打散放在不同的线上。你可以参考图 7-2 的效果。

7.3 范围潜变

项目进行中什么最令人头疼？非范围潜变（scope creep）[①]莫属。说起来似乎我做过的每个项目都在不同程度上受到范围潜变的影响。有时候压力来自挑剔的客户，有时候因为狂热的销售经理想要震惊一下客户。

不幸的是，这就是人生……没有哪个项目的要求是被刻在石头上的。作为开发人员，你需要学会去适应改变。因此，为了让你这个开发人员心里更有谱，我在这里送你一个没有在草图中出现的设计：第二个标签云。

最受欢迎食材标签云

在 Foodbox 中，每个菜谱都列出了所需食材，所以可以多做一个标签云，用来显示最受欢迎的食材。这一次，我会给你自由发挥的机会。

① 也译作“范围潜移”或“范畴潜变”等，指无视时间、成本等因素的情况下改变原有设计或增加新的功能。
——译者注

你要运用从上一个标签云制作过程中学到的技巧来完成这个新的设计。以网格和辅助线为参考来放置标题和其他元素，可以用一些具体的食材，如牛至、蒜、黑豆、苹果、香蕉、奶酪和生菜等，作为标签云的内容。

有三行左右的标签就足够填满侧边栏了。完成这个之后，我们将页面的中间区域填充内容。

7.4 做一个美味的摘要

图片能让页面更显生动。当然，你也可以用大量的颜色，不同的字体或其他东西来为页面增色，但是它们都比不上一张高质量的好图片有冲击力。之所以要强调高质量，是因为低质量的图片会像页面上的一个疮，让人不忍正视。

摄影师是你的朋友

如果想要靠网页设计来挣钱，你可能需要雇用一个摄影师专门为你拍摄图片。这样做的好处多多。首先，你可以不用亲自去学相关的技术，就能获得自己想要的图片。

虽然雇用专业摄影师比较贵，但是他们值那个价。以此为生的人肯定知道拍出好照片的所有技巧。

如果你觉得专业摄影师太贵，或许可以联系一下本地的摄影俱乐部，看看有没有人对此有兴趣。摄影俱乐部里的会员大概都只是业余爱好者，但是通常来讲他们的技术也不错。记住要给报酬。如果你的设计是有偿的，那么他们的服务也应该得到报酬。

不管你打算雇一个专业人士还是一个业余爱好者，你所雇的人一定要是个专家。好比，你雇程序员的时候，是会考虑一个专业程序员，还是某个大学未毕业只想给自己的简历添上一笔的小孩？可能这两种人最后都能完成你分配的工作，但专业人士明显要更可靠一些。

摄影是个技巧活，摄影师需要大量的练习做基础才能拍出好照片。我之所以知道这一点，是因为我自己也曾花过大量时间去拍照片，却连门都没有入。有些客户会交给我一些用便携式数码相机拍的照片，让我把它们放在网页上。那些通常都是畸变的、曝光不足或者曝光过度的照片。虽然用 Photoshop 可以修复一些问题，但是更好的方法还是从一开始就用那些质量好的图片。

那么，从哪里可以找到适合网页用的好照片呢？如果你的预算并不充足，可以访问 iStockphoto[1]这个网站，或者任何其他的图片分享站。甚至在著名的 Flickr[2]上都会有无版权或免费使用的照片。

---

① http://www.istockphoto.com。

② http://www.flickr.com。

假设我在 iStockphoto 上找到了一张意大利面[1]的图标，很漂亮，画面干净而且曝光正常。但是相比我们的页面而言，图片稍微大了一点。没关系，用 Photoshop，我们能从这个照片中裁出好看、紧凑的长方形。

在样式页中你没必要用付费的图片。iStockphoto 中的图片都有水印，除非你出钱买下它们[2]。打开那个图片链接，在浏览器中对着图片单击右键，用复制图像功能将图片放置到剪贴板中。

当把图片放到 Photoshop 中的时候，要调整它的长和宽让图片变大。这种做法对样式页来说是可以的，但是，记住永远不要在一个真实的页面上用这种图片。如果你有可用的照片，那么你应该有高画质的版本，大小应该足够。在客户没有批准方案之前，不要花钱买这个图片。

图片在网上不意味着它是免费的

很多人认为只要一个图片放到了网上，或者出现在 Google 图片搜索的结果中，那他们就能免费地把这些图片用在自己的网站上。这种想法是荒谬的。实际上，图片的版权属于拍摄者，或属于把照片公布在网络上的组织——除非页面上有其他的声明。

在使用任何图片之前，需要得到许可。如果可能，应该拿到一份印刷版的许可，或者是从专业的图片社购买使用服务。

现在把这个 Banner 放到页面上吧，最后得到的结果应该看起来像图 7-3 一样。

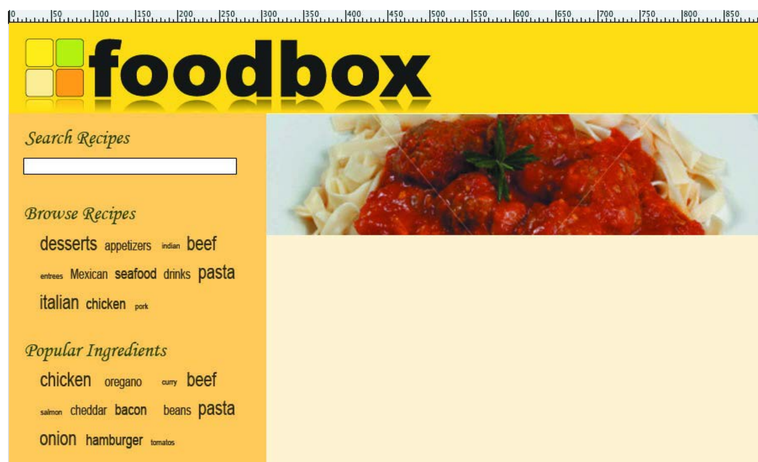


图 7-3 目前的进展

① http://www.istockphoto.com/stock-photo-3762141-italian-meatballs.php。
② 如果在以后你想要买下它也没关系，这张图在 Photoshop 中单独占了一个图层，可以方便地替换掉原来的图。

新建一个名为 Main Group 的图层组，在组里面新建一个图层（Ctrl+Shift+N），取名 Pasta Photo。用 Move 工具将图片的左上角放置 108 像素的水平辅助线和 306 像素的竖直辅助线的交点上。这里是页面的中间区域的准确边界。你之前画的侧边栏和页头的辅助线的相汇于这里。

按下 M 键选定 Marquee 工具，右键单击图片，选择 Free Transform 选项。按住 Shift 的同时将图片的右下角向对角线的下方拖曳，直到图片的右边碰到了整个页面的右边线。按 Enter 键确定变形。

用 Marquee 工具选出图片真正所要显示的地方。从上面说的两条辅助线的交点开始选择，选区应该同辅助线标记出的区域一样大。用向下的箭头移动选区直到选区落在图片中间的部分，确保选区包含你想用的那一部分图片，然后从 Select 菜单中选定 Inverse 选项（Ctrl+Shift+I）。用 Delete 键删除不需要的部分。

最后，用 Move 工具（快捷键 V）配合箭头键将图片移回辅助线划定的 Banner 区域。

7.5　主要内容

除了“Get Cookin’”这几个字之外，我们还不知道在主页上要说些什么，只知道需要在主页上放些文字。虽然我们可以随意拼凑一些文字，但是说不定就会有人对这种做法吹毛求疵。因此，这里会用到一个从传统的印刷业中偷师而来的样式页技术：Lorem Ipsum。Lorem Ipsum 是一段无意义的文字，500 年来都在印刷业中被当作标准。[①]

第一眼看上去，它非常像真实的文本，所以很适合在没有实际内容填充的时候，放在页面的空白区域。去 http://lipsum.org 生成一段文字。这种不知所云的文字非常有用，它让人们的注意力集中在页面的设计而不是内容上——因为任何人都可以一眼看出那些文字根本没有意义。

让我们来给这个页面加点文字吧！用文字工具在页面上写出“Get Cookin’”这几个字，字体是 36 像素的 Monotype Corsiva，颜色是 4B541C。这个文字块应该放在顶部标尺 324 像素和左部标尺 288 像素交汇的位置，它的下端正好是第一个标签云的第二排下端的网格线。

内容文字

建立一个新的图层 Body Text，选择 Text 工具。在左标尺 324 像素和 486 像素的位置设两根辅助线，用来标注文本块的上下边界。在上方标尺 612 像素的位置放置一根竖直的辅助线，确

① Lorem ipsum 是指一篇常用于排版设计领域的拉丁文文章，主要的目的为测试文章或文字在不同字型、版型下看起来的效果。

定文本块的宽。用 Text 工具画一个符合辅助线框定区域尺寸的矩形，将 Lorm Ipsum 粘帖到文本框中。选择这些文字，将行高[①]设置为 18 像素，字体是 12 像素大的 Arial，颜色为黑色（参见图 7-4）。这样文字就嵌入在网格之中了。

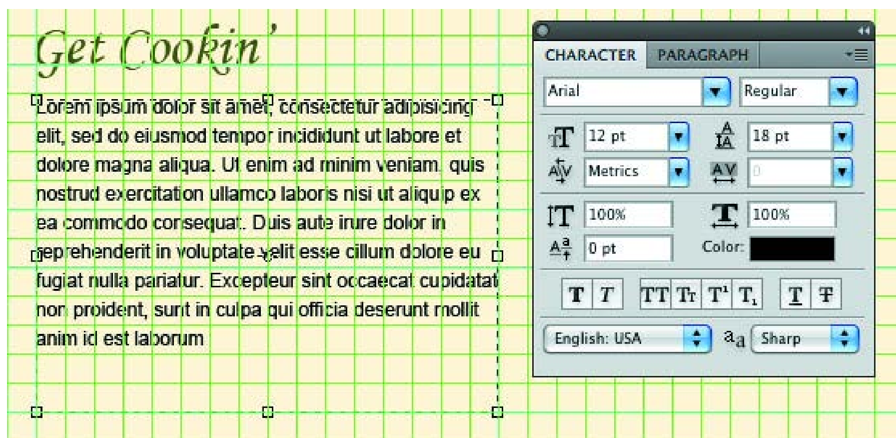


图 7-4 设定行高，让文字和网格平齐

7.6 浏览器模仿

到目前为止，我们建立了一个看起来还不错的页面，但是还没法推断它在浏览器中的样子。现在的这个样式页是定宽定高的，在浏览器中东西是可以“流动”的，所以需要确认一下这个页面在比页面大的屏幕上看起来是怎样的。[②]

从菜单中选择 Image（图像）> Resize Canvas（调整画布大小），将画布的大小从 900 像素调整为 1200 像素。默认设定下 Photoshop 会在原有画布两边均匀地增加像素。

在最开始的设计中，我们决定用定宽的方案，这样会让设计的实现过程变得简单，但是从样式页中看到，在页面的两侧确实有空白的区域。为了让页面看起来更饱满，一个解决方案是将 Logo 和页面中的其他元素居中，然后将背景和页头拉伸至整个窗口的宽度，之前我们也讨论过这个方法。这个方法会让页面有一个全屏的效果，等到代码阶段的时候，再来花更多的时间实现流动性页面的设计。

选用 Move 工具（V），右键单击页头，从弹出的菜单中选择页头（header）图层。这是一个

① 你可能还记得，行距指的是个两条基准线之间的距离（参见第 4 章）。

② 对于以前用纸张工作的人来讲这是一个大问题。纸是有边界的，而网络没有。你需要经常思考自己的设计在无边界世界里的状态。

选择图层的好方法，不用再去图层面板中去苦苦寻找了。选择 Marquee 工具，右键单击页头，选择 Free Transform 工具来激活变形手柄。分别拖曳左右的两个手柄至画布的边缘，按下 Enter 让变形生效。

放大画布后，你可以看到页面最终呈现在屏幕正中的效果。因为你用了 Photoshop 的形状工具来确定页面的不同区域，所以不用担心在变形的时候会出现什么质量瑕疵。

混合布局

近几年宽屏的流行让设计师开始思考新的设计方式来利用屏幕新的长宽比。混合布局指的是页面的多数部分都用的定宽布局，但也有可以随着窗口变化而撑满浏览器宽度的区域。这种布局最近变得很流行。

可以变化的部分通常是页头，主要内容区域还是居中的，就像在设计中显示出来的那样，有时也称之为“可伸展”。这样带来的效果是页头和页脚都撑满了屏幕宽，读者的注意力集中在页面的主要内容上，提高了可读性和组织性。

7.7 小结

在这一章里我们新建了不少元素，也学会了如何模仿显示器中网页的显示效果。在开始将样式页变成代码之前，还有些东西要添加到页面中。开始吧！

第 8 章

样式页上的按钮

Foodbox 上已经有了不少元素了，我们还需要为它加上一个搜索图标，用来做搜索框的按钮。另外还需要在主要区域中添加一个网站注册按钮。

当然，你可以上网找一个带有放大镜的图标作为按钮，或者用一个按钮生成器来做一个。但是，跟你写代码一样，自己做的可调整的东西往往最适合自己的项目。不合时宜的图标可能会毁掉整个网站设计。而且，你用来搜索的时间往往可以用来制作自己的图标——通常只需要几分钟而已。

你可以将这些图片放在之前的那个文件里面，但是为了让文件保持秩序，我们还是在新文件中创造这些元素吧。

8.1 制作搜索图标

制作搜索图标，用 Photoshop 或者 Illustrator 都可以。如果图标要放大缩小到不同的大小使用，那么最好还是用 Illustrator。从制作 Logo 的过程中我们知道，矢量图形可进行平滑的放大缩小，没有质量损失。只是现在我们只需要一个简单的搜索图标按钮，因此用 Photoshop 可以快速地做完这个东西。

新建一个名为 Search Icon 的 Photoshop 文档，长宽都设定为 18 像素，背景色被透明，分辨率是 72dpi。

8.1.1 创建图标背景

按住 Ctrl 和空格键，单击鼠标放大图片。每次单击图片就会放大到另一放大比例，一直单击鼠标直到放大比例是 1200%。这时，你可以看到画布是类似国际象棋棋盘的样子，这说明现在图片的背景是透明色。

将图层的名字从 Layer 1（图层 1）改成 Background，然后在工具面板上选择圆角矩形，将 Radius（圆半径）设置为 2 像素。注意在这里把单位设定为像素是非常重要的，不然 Photoshop 会用默认设定里的测量单位，比如英寸什么的。将工具的模式调整为 Fill Pixels（填充模式），勾选 Anti-Alias（反锯齿）选项。颜色没必要设置，因为稍后我们就会调整它。

对齐像素

我们之前用过 Snap to Grid（对齐网格），这个功能可以约束画出来的形状，只不过现在要做的形状更复杂一点，是带有圆角的。Snap to Grid 可以帮我们确保圆角的每一边看上去都是一样的。为了保证绘制的形状跟像素对齐，单击形状工具选择器右边的箭头，勾选 Snap to Pixels 选项。我建议在制作图标甚至制作整个样式页的过程中，都选中这个选项。

把鼠标放在画布的左上角，按住鼠标不动，将之沿着对角线向右下角拖曳，当鼠标和画布的右下角重合的时候，放开鼠标，形状绘制完成。

右键单击图层的缩略图，选择 Blending 选项。在 Gradient Fill（渐变填充）选项卡里单击 Gradient 风格以编辑渐变的具体样式。把渐变条右端的颜色改成之前标题用的那种绿色，然后把左端的颜色设为黑色（000000）。将这个渐变风格应用到图层上之后，你会看到一个从绿色过渡到黑色的矩形，如图 8-1 所示。

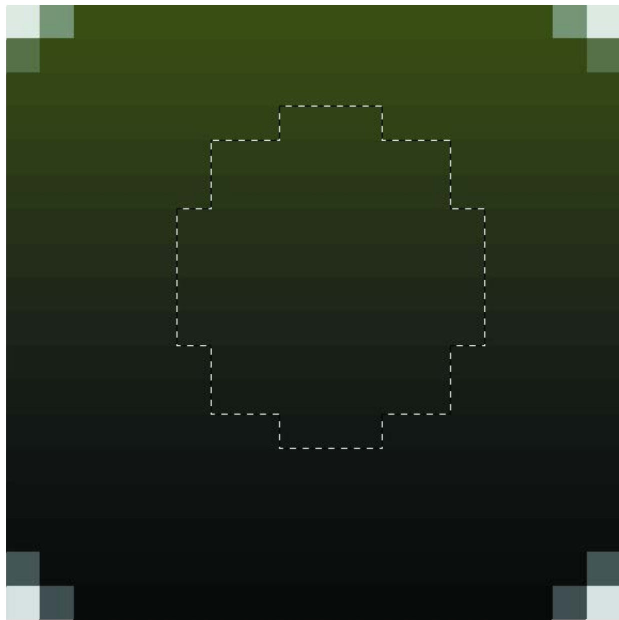


图 8-1　用 Marquee 工具选择出来的圆形选区

8.1.2　绘制放大镜

现在轮到画放大镜了。虽然 Photoshop 提供了一些很不错的形状工具，但是我们会用一种比较古老的 Photoshop 技巧来绘制放大镜：利用它的 Marquee 工具。

创建一个名为 Magnifying Glass 的图层。从工具面板中选择 Elliptical Marquee（椭圆选择）工具。按住 Shift 键的同时，在图标区域内画一个圆形，画出来的效果就像图 8-1 里面那样。在编辑菜单中选择描边，宽度设置为 2 像素，颜色为白色。

选用 Line-drawing 工具，将笔画 Weight（粗细）设置为 2 像素，在放大镜的左下画一条斜线，当作把手。可以参考图 8-2。



图 8-2　带把手的放大镜

现在，我们需要往上面加一点玻璃的效果。虽然选择工具可以选中刚刚画的圆圈内部，但是那太费事了，用 Magic Wand（魔杖）工具选择圆圈会更容易些。从工具面板中选择 Magic Wand 工具，在圆圈内部单击一下，这样这块区域就被选中了。

新建一个叫做 Glass 的图层。将 Foreground Color（前景色）设置为白色（FFFFFF），选中 Gradient Fill 工具。在 Options 工具栏中双击渐变图案，预置模式中选择 Foreground to Transparent（前景色变透明）。设置好以后在放大镜内部从上到下画一条直线，记住按住 Shift 再画，可以画出竖直的直线。

完成上面的步骤后，你应该得到一个类似于图 8-3 里的图案，保存 Photoshop 文档，命名为 search_button.psd，跟其他的 Photoshop 文件一样放在 originals 文件夹中。



图 8-3　完成的图标

8.1.3　放置搜索图标

切换回样式页的文档，单击 File 菜单下的 Place 选项，选择刚才创建的 search_button.psd，将它放置在搜索框的旁边。摆放好之后，保存文件。图 8-4 是完成后的搜索区域的效果。



图 8-4　完成的搜索区域

8.2　创建注册和登录按钮

在草图中，页面的右侧有两个很大的按钮。在样式页的空白处创建这两个按钮，我们需要用

到在 8.1 节中用到的一些技巧。新建一个文档，名叫 button。背景色为透明，宽 216 像素，高 72 像素。这样的尺寸可以很好地融入到样式页中的空白空间。

选择 Rounded Rectangle（圆角矩形绘制）工具，将圆角半径设置为 10 像素，这样我们的按钮就会有漂亮的圆角。

从图层面板中建立一个新图层，或者用快捷键 Shift+Ctrl+N。取名为 button background。我们会在这个新图层上画出按钮。

用刚刚设置好的圆角矩形画出按钮。具体做法是：从画布左上角开始按下鼠标拖曳至右下角。如果你设置的前景色不是白色，画出来的效果应该更清楚些。但是无论什么颜色都不要紧，马上我们就会更改颜色了。

然后右键单击有按钮的那个图层，选择 Blending Options。在弹出的 Layer Style（图层样式）窗口中选择 Gradient Overlay（渐变叠加）。双击窗口中的渐变图案打开 Gradient Editor（渐变编辑器）。如图 8-5 所示，颜色桶用数字 1 标出，渐变点则是 2。

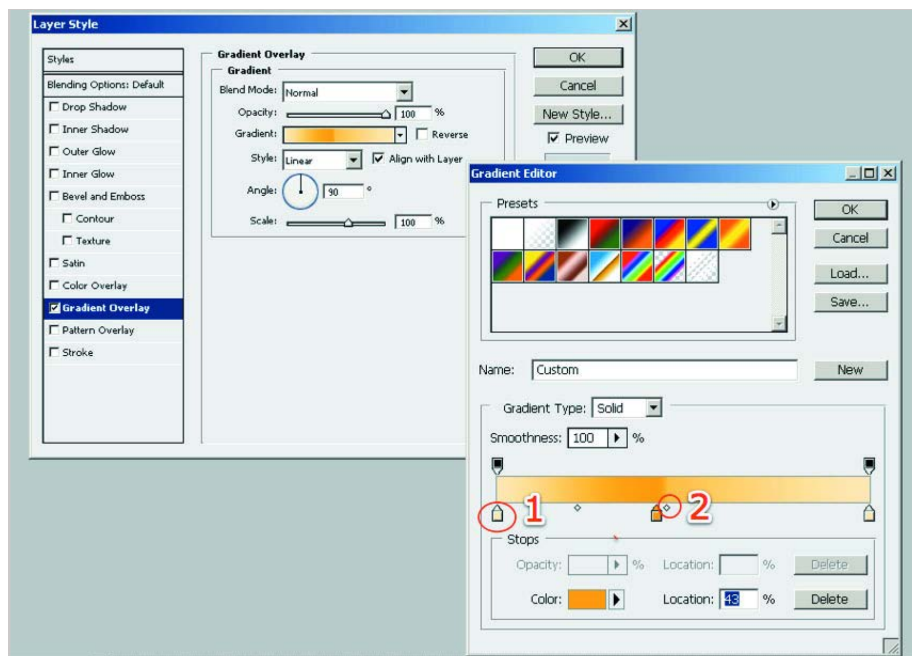


图 8-5　渐变选项

我们需要一些设计，好让按钮具有可识别性。将渐变画好的关键在于控制颜色在渐变中的变化程度。渐变编辑器中有一个色带，用以显示当前的渐变图形。

单击颜色桶来设定渐变中的颜色。当你第一次打开 Gradient Editor 的时候，渐变色带上默认只有两个颜色桶，单击颜色桶之间的空白地区可以增加颜色同的数量。

按钮只需要两种颜色，但是却需要三个颜色桶来设置过渡。将最右端和最左端的颜色桶都设为 FFEABF，然后在色带下方两个颜色桶的中间单击一下鼠标，添加一个颜色桶。这个颜色桶的颜色是 FFAE00，是侧边栏中的橙色变化出来的颜色。

注意编辑器中渐变带下方的小圆点。向不同的颜色桶移动这些小点，可以增大或减小颜色在渐变中的混合程度。我们要的按钮需要一个比较突然的过渡，所以让右边的那个点尽可能地靠近中间的颜色桶。这样按钮中间就会出现一条水平的线，让其显得有立体感。在图 8-5 中可以看到我给出的具体设置。设置好渐变后单击 OK 按钮。你也可以自己动手试试这个编辑器，图 8-6 里就是一些不同的效果示例。

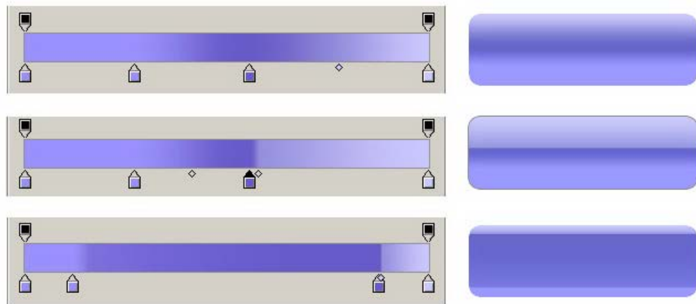


图 8-6　用渐变来创造立体的视觉效果

在按钮四周加个边框能更加突出按钮。在图层效果中选择 Stroke，笔画宽设为 1 像素。用黑色或者类似的暗色可以突出按钮，而浅色的边框则让边框不那么明显。可以把透明度降到 45%来软化边框。记住要选择内边框，不然边框会改变按钮的长宽（参见图 8-7）。

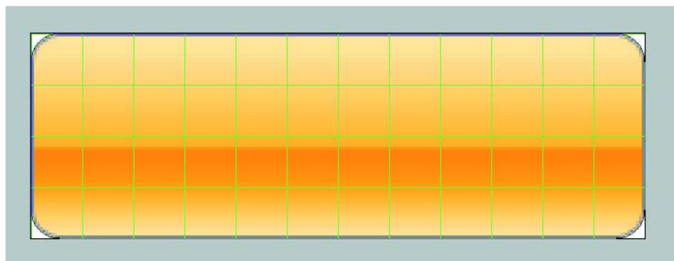


图 8-7　完成的按钮

将按钮存成 button.psd 文件，放到 originals 文件夹下，待会儿你需要它来创建登录按钮。

8.2.1 添加文字

选用 Text 工具，在按钮上加上“Sign Up”这几个字，你可以根据颜色理论，试试不同的颜色。我建议给文字使用一种比较暗的颜色，不管用什么颜色，都要注意让文字与按钮背景明显地区分开来。也可以试试给文字图层加上一些效果让它显得更立体。

将这个文件存为 signup_button.psd。

8.2.2 添加注册按钮

切换到样式页文档，用放置 Logo 的方式将按钮导入进文档。将按钮放置在页面中为其预留的位置上。

记得保存文件。

创建登录按钮

重新打开 signup_button.psd 文件，将文字改成“Log In”。选择 Move 工具，调整文字的位置让其在按钮上居中。之后保存文件，命名为 login_button.psd，将文件导入样式页中，放在注册按钮的下方。

在图 8-8 中可以看到摆放好的按钮。

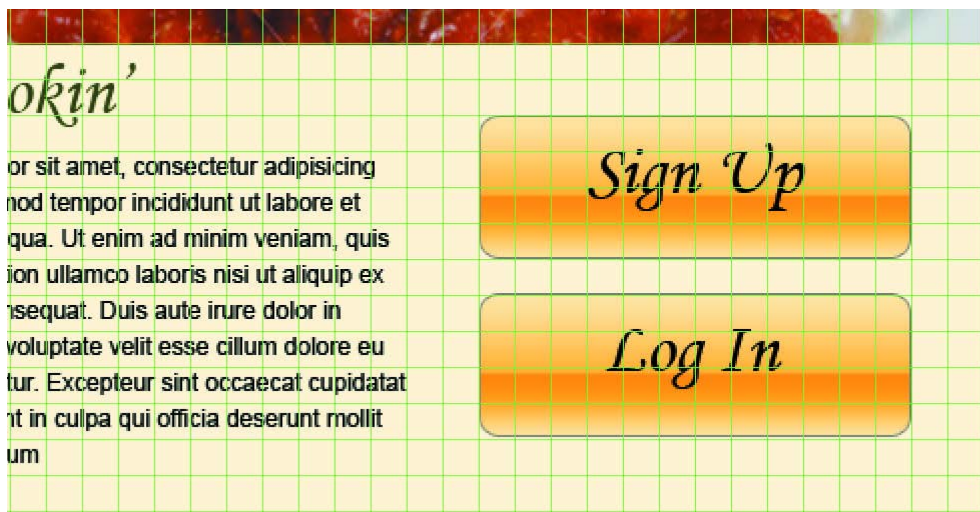


图 8-8 两个按钮就位

8.3　文字内容来了

Steve 告诉你，老板们终于确定了在主页上要放的文字了。他们很喜欢页面现在的样子，但是也察觉到了整个页面还有不少的留白，所以他们决定保留原来那个版本中新添加的菜谱出现在首页上的功能。他们想在首页上摆放的文字如下：

Foodbox is the best way to collect and share recipes with the rest of the world. You can build your own recipe book from thousands of great recipes from renowned chefs or users just like you. You can also share your own secret recipes with a few of your friends or make them available to the rest of the world!

Create an account today and get cookin'!

8.3.1　替换掉原来的乱码

替换文字的工作很简单。将上述文字放在页面中的文字区域中，删掉之前的那段乱码。但是你会发现这段文字可能比原来那段要短，所以我们需要做一些微调。选中刚刚粘贴到页面上的文字，将字号改成 14 像素。可能这样做还不足以占满之前那段文字所占的空间，但是你现在有了多余的位置来放老板们要求的菜谱。

8.3.2　添加“最新菜谱”区

其实我们不必填满页面下方的所有空间，用一个标题加一些文字块来显示菜谱就可以了。做一个新的 16 像素的标题，标题的内容是“Latest Recipe”，字体是 Monotype Corsiva，跟其他的标题一样。将这个标题放在那段介绍文字的下方，记得在它们之间空出一些空间。你可以复制“Get Cookin'”图层，调整文字的位置，这样就跳过了选字体、字号和颜色几个步骤，可以快速地做好这个新标题——这就是所谓的捷径。

在刚刚做好的标题下面加上一些假的菜谱，这次我们用14像素大小的 Arial 字体。让菜谱文字有个稍微靠右的缩进——18像素，也就是一整个网格单位的距离，可以参考标签云的效果。每个菜谱自成一行，每一行之间都要有一个行高的距离，用网格线作为参考。

在菜谱下面为其加上描述，用 12 像素大小的 Arial 字体，同样给它 18 像素的缩进。在图 8-9 中可以看到效果。

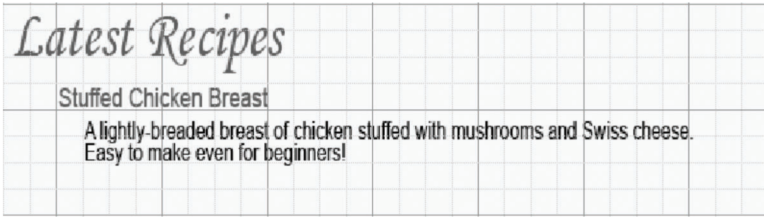


图 8-9 “最新菜谱”区

8.4 小结

现在，你可以创建一个真实的页面了。在这部分中，你学会了如何用图层组工作，如何处理文本，如何运用蒙版和一些图层效果。这些技巧在今后的页面设计过程中是非常重要的，因为每次你都需要向客户展示一个醒目且吸引人的演示页。最后，当你需要搭建一个最终版页面的时候，这个样式页会给你提供切实可用的参考（参见图 8-10）。



图 8-10 完成的样式页（另见彩插）

Part 3 第三部分

建设网站

本部分内容

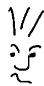
第 9 章

用 HTML 做出主页

你已经在设计页面上花了大把的时间了，现在要做的是将这个样式页转换成功能齐全的网页。这个过程需要经过几个步骤。先要考虑以什么样的结构来组织内容，其次要写出 HTML 的框架，用简单并且结构化的标签来定义页面上的区域。最后，需要用 CSS 来为内容添加上基本样式布局。同样地，你还需要为页面添加颜色、字体样式和图片。

在这一章中，我们先要架起 HTML 文档的框架，然后在接下来的三章中，会研究如何用 CSS 来实现页面上的各种设计。

网页跟 Photoshop 文档中被固定的画布是不一样的，所以有时转换的时候会出现偏差。跟我们通常看见的插图小册或者海报不同，网页的大小是会变化的，因为浏览器的显示区域会随着屏幕大小或是窗口大小的改变而变化。比如，用户可能会用一个很小的窗口来查看网页，也可能会将窗口最大化到跟屏幕一样的长宽来浏览。因此，你几乎需要在制作过程中不断地做出调整来满足这些条件。虽然完成的页面跟样式页看起来不会一模一样，但是它们看起来应该是几乎差不多的，大多数人都不会觉得网页跟样式页有什么不同。所以，不要纠结于如何让网页和样式页完全相同——这很重要。



小乔爱问……

我已经掌握 HTML 了，还需要继续看完这章吗

你可能已经熟悉 HTML 了，并且知道浏览器呈现页面的基本概念，但这还不够。想想当初你花了多长时间才能写出优良的代码。要知道，需要学习的不仅仅是 HTML 的语法，还包括这些语句背后的原理。如果不了解其背后的原理，那么你就很难搞定样式表；将它实现成可以跨浏览器正常浏览的网页就更困难了。本章内容将引导你去理解 HTML，而不是浅尝辄止。你会从单纯地手写 HTML 代码，进步到最后用后台程序生成 HTML。

9.1 网页标准化

程序员总在考虑“分离”这个问题。开发网络应用的时候，通常会涉及模型（model）、控制器（controller）和视图（view）三个部分，作为程序员，将显示层的逻辑和业务层的逻辑分开，是一个很好的习惯。网页设计师对这个理念也有很好的理解。一个按照网页标准设计的页面，会把页面内容从设计和行为中剥离出去。

Web 标准化通常是指使用标准化的方法和理念来制作网页。像 W3C（World Wide Web Consortium，万维网联盟）这种制定标准的联盟会出台大量相关的标准。另外，网页设计社区中的先头兵们使用的好方法和形成的好习惯，也会慢慢地变成标准。

遵从网页标准化的网站和页面通常有以下几个特性。

- 用合法的 HTML 或者 XHTML 标签来组织网页的结构和内容，其中包括正确的 doctype 设置和编码设定（character set）。
- 用合法的 CSS 来呈现页面样式，这意味着页面的布局、文字颜色、字体、页面颜色和其他任何非内容的元素都是用 CSS 来控制的。
- 任何人（包括残障人士）在任何平台上用任何浏览器都可以访问这个网站。
- 导航、链接和结构符合基本的可用性规范。
- 页面动作跟内容和样式是分离的。JavaScript 部分应该可以在所有平台上正常运行，在不支持 JavaScript 的设备、平台上，或是面对那些无法使用 JavaScript 的用户时，页面的基本功能仍然正常。

这些要求听起来很合理，但是到底要怎么实现这些要求呢？本章会从 HTML 文档开始——写一个合法的 HTML 文档来组织内容和页面结构。如果你有过一些 HTML 的相关经验，这部分对你来说是小菜一碟。即便你从来没和 HTML 打过交道，也不必担心。跟随着本章的课程，你能很快地学会。

9.2 首页的结构

试着将页面想象一块块由内容组成的区域，而不是行和列的组合。这种角度会帮我们开发出既符合标准又有灵活性的页面——即便在切断样式表的关联之后，还能改变页面的布局。

对于 Foodbox 来讲，我们希望侧边栏和主区域的内容都能各居其所。所以，要把刚刚做样式页时做过的事情再来一遍：将页面划分成不同的区域。

基本上，需要将样式页划分成如下几个区域：

- 页头
- 侧边栏
- 主要区域
- 页脚

这四个区域是很容易就能看出来的。然而，如果想要建立一个更加灵活的结构，以便可以更轻易地控制它，你还需要在这几个区域下面划分出几个子区域。为了达到这个目的，需要找出内容和内容之间的逻辑联系。

为了举例说明，我们将样式页的区域用大纲表现出来：

- 页面
  –页头
  –页中
    * 侧边栏
      • 菜谱搜索
      • 菜谱浏览
      • 最受欢迎食材
    * 主要区域
  –页脚

在上面的例子中，最外面的一层是页面，然后我们将这一层分成了页头、页中和页脚三个区域。这个叫做页面的外层区域（或父区域）的部分，可以作为定位页面内元素的参考。同时，页面总宽度也可以通过改变这个父区域的宽度来调整。

侧边栏和主要区域被一个叫做页中的容器包裹着。同样的，这个页中区域也可以作为参考点，但是它还扮演一个更重要的角色：提供灵活性。可能在有的页面上，不需要侧边栏，而需要把整个页中区域都用来显示主要内容。那么在这些页面上，我们就可以去掉侧边栏区域，把内容都放在主区域中，然后用 CSS 来调整它的大小。

这种结构其实是很常见的，是一种标准的带页头和页脚的两栏布局结构，也是在网页中最常见的布局方式。标准化的设计带来的好处是，如果你愿意，网页框架可以在另外一个项目中重复使用，因为那些跟列宽、配色和其他视觉相关的元素是单独存储在样式表中的。[①]

---

① 这种方式对于设计网站的外观非常有帮助，你可以使用这种技术让用户选择喜爱的主题。访问 http://www.csszengarden.com 站点，这里面有一个很强大的示例，演示如何以不同的方式呈现同一个文档。

9.3 语义化的标签

语义化标签组织的文档能同时被机器、设备和人读懂。例如 Google 的网络爬虫会利用链接中 `h1`、`href` 或类似标签来确定网页和网页上内容的优先级。



小乔爱问……

为什么不对样式页进行切片

在网络开发的早期——指的是 2004 年以前的“中世纪”，用 Fireworks 或者 ImageReady 一类的工具来对 Photoshop 文档进行切片是开发者的常用手段，切片好的文档可以转化为 HTML 文件。这种手法虽然快速但质量不高，因此也引出了不少严重的问题。

譬如切片得到的 HTML 总是用 `table`（表格）标签来进行布局，虽然在使用 CSS 之前，这是大多数网页设计师采用的技术。这种技术带来的问题也有很多，比如说给使用屏幕阅读器[①]的用户造成了极大障碍。

同时，这种方法也没有把内容和样式分离开，所以你没法便捷地调整页面内容的显示样式让它具有更强的适应性。这样就无法很好地满足各种需求，如打印机友好页或者是适合移动设备显示。

最后，也是最重要的一个原因是，如果使用了表格布局，那么 HTML 代码中关于表格的代码将不得不在每一个页面上都被重复一遍。每次用户发出一个页面请求，这些数据就会被传送一次。对于规模比较小的网站来说，这会导致页面呈现到用户面前所需的时间变长；对于访问量大的网站来讲，你会从 ISP 每个月寄给你的账单中看到表格布局对你的影响。因为架设一个网站，你需要为网站产生的流量付费。如果网站的数据量过大，那么你应该对任何可以减小网站体积的手段感兴趣。

用 CSS 实现的标准化的网页设计，那些关于网站外观的文件只需要被用户下载一次，而且所有页面都是共用同样的文件，这样可以改善网站的性能，同时节省费用。

要有目的地使用 HTML 标签来表述标签标记的内容。页面有标题、段落、列表和其他元素；相应的，HTML 中设计了各式标签来标记不同的内容。以标题为例，可用`<h1>About Us</h1>`之类的标签来标记它。HTML 解释器能读懂这个标签，知道它标记的是页面上最重要的一个标题。

---

① 屏幕阅读器是一种软件，用来将文字、图形以及机器界面的其他部分转换成语音。——译者注

但是，用`<fontsize="+2">About Us</font>`这种标签来标记标题则是极不恰当的。然而很多程序员恰恰就是这样做的，因为他们不喜欢`<h1>`标签会在被标记的文字上面加上一个间距，并且会在标签被呈现的时候自动换行。[①]

其实，用CSS就可以解决这些视觉上的问题，前提是你将这些东西背后的原理弄清楚。具体说来你可以做的有：用CSS来改变所有标题的样式，或是针对某一页上的某个特定的标题来做出演示变动。更令人欣喜的是，一个CSS文件可以对应多个页面，这只需要像一个CSS文件里加入几行代码就好了，而不需要你手动去修改 100 个页面上的所有标题。

9.4 主页的框架

打开你最常用的文本编辑器[②]，新建一个文件，然后将这个空文件保存成 index.html。这个文件将会是我们网站的主页，网站服务器如果收到一个不带任何特殊页面指向的请求，就会返回这个页面。

9.4.1 doctype

每一个 HTML 页面都需要指定一种 doctype[③]，这样，验证工具才能判定页面中的标签是否书写正确。而且，保证页面合法是非常重要的，在开始写 CSS 和 JavaScript 之前就需要确定合法性的问题。因为不合法的标签会导致 CSS 的呈现错误，或者是让 JavaScript 发生严重的问题。你的浏览器也需要有一个完好的页面来加载样式和动作，所以一个未闭合的标签可能会导致浏览器出错。

更重要的是，doctype 可以强制某些浏览器以不同的方式解析页面。例如，IE6 会以一种诡异的模式容忍不严格合法的标签，让你不得不抓着脑袋痛苦地调试页面，让它可以在那些严格遵守页面规范的浏览器中显示出一致的样式。你可以利用 doctype 声明来强制 IE6 以标准的模式呈现页面。虽然效果还是不太尽人意，但是也勉强可用。

你可以从几种不同的 doctype 选择一种，选用的类型决定了你在页面上能使用哪些标签，同时也决定了检验页面上标签是否合法的衡量标准。最常见的两种 doctype 是 XHTML 1.0 Transitional 和 HTML 4.01 Strict。

---

① 有些所见即所得模式的 HTML 编辑器都是这样写的代码，所以这并不是只有初学者会犯的错误。

② 推荐 Windows 用户使用 Notepad++，Mac 用户使用 TextMate。

③ Document Type（文档类型）的简写。——译者注

默认页面的名字

在网站服务器中有一个概念叫做默认页面。当发送到服务器的请求没有指定任何目录下的页面时，默认页面就会被传送给用户。在服务器里，文件以目录结构组织起来，页面分别存放在不同的文件夹之中。URL（Universal Resource Locator, 统一资源定位符）中则包含了用户所请求的文件或者页面的路径。例如你发出了对 http://www.foo.com/products/superwidget/about.html 的 URL 请求，位于 http://www.foo.com 的服务器会在 products/superwidget 文件夹里搜寻一个叫 about.html 的文件。

但是，如果你请求的 URL 是这样的：http://www.foo.com/products/superwidget。这就是一个不完整的源，那么服务器就得先弄清楚这条 URL 的意思。首先，服务器会先看看这个地址到底是个什么东西。如果发现这里是个文件夹，服务器就会将文件夹里的文件名同一个默认页面文件名列表对比，查看文件夹里的文件名是否出现在这个列表中。常见的默认文件名有 index.html、index.htm 和 default.htm。

要是服务器没有找到默认文件，那么它可能会返回一个目录列表。如果管理员配置了服务器设置，禁止了目录列表显示，那么就会返回错误信息。很多管理员认为这样做可能增强网站的安全性，我却觉得即便这样做了也不会有什么大改变。如果有东西是你不希望别人看到的，那么别把它放到网上。

想要链接一个有默认页的资源，要么在 URL 里写上文件的名字，要么在 URL 中文件夹的最后加上斜杠。用这种默认的 URL，就相当于告知服务器你实际上在请求一个文件夹，并且希望服务器能返回这个文件夹里的默认页。默认 URL 最适合用在页面首页上。

为了效率最大化和避免混淆，你应该链接完整的资源信息。Foodbox 的首页链接就应该总是以 index.html 结尾。这样，服务器就只返回这个文件，并且紧接着就能处理下一条请求了。

1. XHTML 1.0 Transitional

XHTML 1.0 Transitional 在很长时间都被认为是做网页的正途，其中一个主要的原因就是这种 doctype 会强制浏览器进入标准模式。虽然这个问题在今天看来已经不再是个问题了，但是 XHTML 仍然有不少优于普通 HTML 的地方。XTHML 更加严格，这让开发者更多的思考关于页面结构的东西。同时，它还规定了标签中只能用小写字母，这对解析页面有利。并且，XHTML 还要求每一个标签都有一个相应的闭合标签。

然而一些跟浏览器有关的问题给这些优点带来了一些阴影，比如扩展性。在 IE 中，如果把 content type 设置为 `appliction/xhtml+xml`，它是无法正确处理 XHTML 的。只有把 content type

设置成 `text/html` 的时候，IE 才会解析 XHTML，而且把 XHTML 当成 HTML 来解析的。显然把 content type 设置成前者要合理一些。在 IE 之中将 doctype 设置成后者后，本来是等待解析 HTML 的浏览器得到却是 XHTML 标签，这让浏览器不得不做一些转换工作。[①]这样你就会失去原本属于 XHTML 的优点，并且这些跟浏览器相关的问题还会导致新的毛病。例如自闭合的 `div`、`span` 标签在 XHTML 中是完全没有问题的，但是当浏览器将其当做 HTML 解析时，自闭合标签中的后置斜杠被拿掉了，留下一个没有闭合标签的 `div`，打乱了这个标签之后所有的结构。[②]

这些问题驱使一些设计师和开发人员回到了普通 HTML 的怀抱，他们通常遵循 HTML 4.01 Strict[③]或是 HTML 5 标准。

2. HTML 4.01 Strict

我们会用 HTML 4.01 Strict 来写本书的例子。在 HTML 4.01 Strict 中，元素仍然是层次化的，但是大小写不再是问题，一些标签页不需要被闭合，同时自闭合标签是不被允许的。注意，这些只是语句本身的一些差异，并不是说 HTML 在句法上要优于或者逊于 XHTML。只要你注意了对文档本身合法性的验证，浏览器兼容、用户体验、CSS 或是 JavaScript 什么的都不会是问题。

虽然这里只会用到 HTML 4.01 Strict，但我仍然会强调写出结构良好、合法且有语义的标签。这样可以确保以后转换到 XHTML 1.0 Strict 或者 HTML 5 的时候不会有什么障碍。记住，不管你最后使用了什么样的 doctype，最终浏览器所面对的总会是 HTML，所以这两种 doctype 只是在句法上有些区别。大可不必在这个问题上太过纠结。

3. 添加doctype

在文档中置入如下的 doctype 声明。文档中所有的其他内容都应该写在这句声明之后。

homepage_html/index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

你并不用费事将上面的声明一个字母一个字母地敲出来，大多数网页编辑器都提供了可用的模板，或者你可以用搜索引擎搜索“HTML 4.01 Strict doctype”这几个关键字来得到一个样本。

---

① http://xhtml.com/en/xhtml/serving-xhtml-as-html/。

② http://www.webdevout.net/articles/beware-of-xhtml#myths 上有很多非常好的示例，说明了 content type 如何呈现以 XHTML 编写的网页。

③ http://mezzoblue.com/archives/2009/04/20/switched/。

9.4.2 html标签

一个网页其实和 XML 文档类似，是一种层次化的元素合集。其中 html 元素是文档中的根元素，所有其他的元素都是它的子元素。几乎所有 html 元素都由一个开始标签和一个闭合标签组成，它们标记了这个元素的范围。你可以将开始标签和闭合标签想象成 Java 中的大括号。

紧接着 doctype 声明之后，写上 html 标签，别忘记要把闭合标签也写上。时刻记着闭合标签是刚开始做网页开发时候需要养成的好习惯。写一个标签，马上再加上它的闭合标签，最后再把光标移动到两个标签之间添加内容。忘记闭合标签会造成标签不合法，进而导致浏览器会胡乱解析页面样式。不合法的标签还会引起其他开发者的不满（他们会因此而揍你一顿也说不定）。你要尽可能避免这种事情的发生。

homepage_html/index.html

```
<html lang="en">

</html>
```



小乔爱问……

XHTML 走到尽头了吗

W3C 今日决定停止下一代 XHTML 的开发，集中资源主攻 HTML 5①。但是这并不意味着 XHTML 1.0 走到了尽头。这只说明表示如果是做网页标签，HTML 5 是正确的选择。

多数程序员和标准化支持者偏爱 XHTML 的原因是，它更严格：所有的标签必须有闭合标签，所有的标签和它们的属性都需要用小写字母书写，属性的值必须要用引号标记，并且独立标签如 br、img、meta 和 hr 标签中需要有后置斜杠。除了自闭合标签之外，以上所有的特性在 HTML 4.01 Strict 都是合法的；在 HTML 5 中你则可以用到上述所有特性。

XHTML 的研发终止了，所以我们可以说它跟 COBOL 的死法一样——它还能用，只是停滞不前了。所以没有必要急急忙忙把你现在所有的页面都变成 HTML 4.01 Strict，但是当开发一个新网站的时候，你就需要综合考虑一下所有选择了。

9.4.3 属性

每个标签都有各自的不同属性，在标签声明中可以指定某些属性，这些属性让标签的定义更

① http://www.w3.org/News/2009#item119。

加具体。html 标签就有个用来表述文档语言的属性。

自闭合标签

如果你熟悉 XML，关于自闭合标签的概念你不会陌生。它们就是那些有后置斜杠的标签。HTML 4.01 Strict 不支持这种特性，但是 XHTML 1.0 Strict 和 Transitional、HTML 5 都支持它。

9.4.4 head和body标签

在 html 标签中总是会发现这两个元素的身影：head 和 body。head 标签包括了所有的元数据（metadata），像在浏览器标题栏和收藏链接中的显示文字，JavaScript 文件的链接，样式文件和其他有用信息。body 元素包含的内容将出现在网页中。

紧接着 html 之后，在你的文档中加上 head 标签，别忘了闭合标签。

homepage_html/index.html

```
<head>

</head>
```

就像写程序一样，对 HTML 代码做适当的缩进也是很有必要的。特别是当你的文件很大时，缩进在开发后期会起大作用。

在 head 元素中加上以下内容：

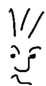
homepage_html/index.html

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Foodbox</title>
```

9.4.5 没有闭合标签的标签

HTML 中有些标签是没有范围的，因为这些标签既不包含任何内容，又不会对页面内容产生影响。因此，这类标签本身常常被看作是页面的内容。

具体说来，img 标签就是个例子，它用来向页面中插入图片；另外还有 br 标签，用来换行；还有 hr 标签，它是一条水平分割线。


小乔爱问……

不是应该在 HTTP 首部中设置内容文字和编码方式吗

正确设置 HTTP 首部是很重要的，但是有些浏览器和验证器是以 `meta` 标签中的内容为准的。在页面源中使用 `meta` 标签能够更好地描述网页，并且能让其他开发者更清楚地了解你的意图。

最重要的是，`meta` 标签能够在本地开发和检测 HTML 文件，从自己硬盘打开的文件可以按照设定的编码正确呈现。

当你在服务器上部署 HTML 文件的时候，要注意保持 `meta` 标签中的编码设置同 Content-Type 首部的编码设置是一致的。

`meta` 标签是一个内容元素，用以让我们以元数据的形式描述文档。在个例子里，我们用 `meta` 标签告诉浏览器和解释器页面内容的编码方式。有时，你会从其他来源复制一些内容粘贴到文档中，这些被粘贴过来的内容可能包含了部分浏览器和机器无法识别的文字符号。指定一种编码可以让 HTML 验证器在发生这种情况的时候提醒你。

`meta` 标签还可以给浏览器、搜索引擎和其他网页客户提供更多信息。在第 18 章中我们会接触到更多关于这个标签的内容。

9.4.6 页面标题

`title` 标签很重要。在这个元素内的文字会显示在网页浏览器的标题栏上。同时，如果网页被收藏，它还将是收藏夹里这个网页的默认存储名。最后，大多数搜索引擎的返回结果都会包含页面标题。在这个例子中，网站的名字已经够用了，但是在子页面上，注意给这个元素加上额外的文字，如 About This Site、Foodbox or Top Recipes、Foodbox。因为这个标题会出现在浏览器标题栏和收藏夹中，所以我们希望网站名字总是在里面。但是，这个标题有被删减的风险，所以我们希望能让某一部分最先显示。对于用户和搜索引擎来说，Latest Recipes、Food…这样的标题看上去要比 Foodbox、Latest Rec…好很多。

随着产品接近尾声，页面的 `head` 区域会包含更多内容。现在，你可以开始着手创建页面的视觉内容了。在这个阶段，还没有必要做太多的搜索引擎方面或脚本编写的事情。

块元素和内联元素

body 标签中的元素不是属于块元素，就是属于内联元素。了解这两种元素的区别能够让你在写 CSS 页面样式的时候节约不少时间。

一般说来，块元素要另起一行才能开始，如 div、h1、h2、h3、p、ul、li、table 和 form。

反过来，内联元素则是跟其他元素在同一行内被呈现的，它包括的标签有 a、b、i、span、em、strong、label、select、input、textarea、u 和 br。

你需要牢记的是：块元素可以包含其他的块元素或者内联元素，而内联元素中则只能包含内联元素，块元素不能被放在内联元素中。①

9.4.7　body标签：重头戏

页面内所有的可见内容都坐落在 body 标签中。

在文件中写上 body 标签并将之闭合，在两者之间留出一点空间写主体内容。这样，我们就有了一个标准的 HTML 4.01 Strict 模板了（参见图 9-1）。

homepage_html/index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">

<html lang="en">

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Foodbox</title>
  </head>

  <body>

  </body>
</html>
```

图 9-1　一个预设的 HTML 模板

在 9.2 节中，你已经学会了用不同的元素将页面分成不同区域。现在要做的是用代码将这些部分划分出来，这时需要用到的标签是 div，它可以起到划分的作用。div 是不可见的元素，在

① 在内联元素中的块元素可能会被呈现，但是整个页面就变得不合法了，而且这会给样式表的应用和写 JavaScript 的工作带来问题。

页面呈现的时候不会占据页面空间。但是它也有比较重要的特性。首先记住它是个块元素，这意味着它会另起一行。接下来你会见到关于块元素的更详尽解释。

1. 页面容器

先建立一个顶层的区域，然后可以用这个区域来包裹住页面上的所有内容，侧边栏、页头、页脚等都会放置在这个区域中。而且，我们还可以利用这个区域保证 900 像素的宽度。稍后，这个区域还将充当其他所有元素的参考位置。好的程序员都会为程序写注释，HTML 也提供了注释功能。在 body 的开始标签后加上以下代码：

homepage_html/index.html

```
<div id="page"> <!-- start of the page wrapper -->

</div> <!-- end of the page wrapper -->
```

你还需要一种辨识符来告诉浏览器某个区域需要某种样式和动作。注意，id 这个属性在整个文件中都是唯一的，也就是说在一个页面上只能有一个 id 是 page 的区域。如果超出一个，页面将通不过验证，同时样式方面还会出现诡异的问题。

随着时间推移，HTML 文件会变得冗长，且不易阅读，上面写的注释可能在以后会有帮助。

2. 四个内容区域

用 div 元素划分出页头、页脚、侧边栏和页面主要区域：

homepage_html/index.html

```
      <div id="header"> <!-- start of header -->

      </div> <!-- end of header -->
      <div id="middle"> <!-- container for the sidebar and main region -->
        <div id="sidebar"> <!-- the sidebar -->

        </div> <!-- end of the sidebar -->

        <div id="main"> <!-- start of main content -->

        </div> <!-- end of main content -->
      </div> <!-- end of middle container -->
      <div id="footer"> <!-- start of the footer -->

      </div> <!-- end of the footer -->
    </div> <!-- end of the page wrapper -->

  </body>
</html>
```

示例代码中有一个新加的区域叫做 middle。通常，每当在页面上有两个区域需要并排显示的时候，你都需要用另外的一个区域将它们包裹起来。这样可以在不添加太多标签的情况下，提供更大的灵活性，因为你可以只对最外层的那个区域做样式调整。在上面，将侧边栏和主要区域用容器包裹起来，就像用容器包裹起整个页面一样。

现在结构搭好了，可以添加内容了。

默认文字

图片标签的 alt（alternative 的前三个字母）属性提供了一种改善可用性和可访问性的简单方法。当图片无法显示的时候，默认文字（alternative text）就会填补空缺。盲人其实就是靠这些文字来了解页面上的图片内容的，所以应该尽量让默认文字表述得清晰详细。“一辆蓝色的汽车”就不如“一辆 1957 年的复古雪佛兰汽车停在市中心的商场门口”来的详细。

默认文字也是基于文字的浏览器和低带宽手机用户的福音。另外，搜索引擎也利用这些文字来查询网页，这也是需要重视默认文字质量的一个原因。搜索引擎无法读图，默认文字就显得尤为重要。在 16.2 节中，我会讲到更多的相关内容。

9.5　页头

页头部分只包括 Foodbox 的 Logo，我们会用 img 标签来添加这个图片。img 标签有属性用来指定图片的路径，类似于 a 标签中的 href 属性。这个属性的值可以是个 URL，也可以是图片文件的相对路径。我们会在 9.6 节中详细讨论关于 URL 的问题。

往页面上放置图片的时候，最好能指定它的长和宽。虽然现在我们还没准备好图片，长和宽可能都得空着，但是过会儿还是要加上的。现在，还是先给图片指定源和 alt 文字属性。默认文字可以在图片不能加载的时候显示，同时也为使用屏幕阅读器的用户提供极大的方便。

把鼠标光标移到 id 为 header 的 div 标签之后，插入如下代码：

homepage_html/index.html

```
<img src="images/banner.png" alt="Foodbox">
```

保存文件，页头部分暂时就做好了，我们继续接下来的部分。

9.6　侧边栏

侧边栏区域中内容不少，包括了一个搜索区域、一个菜谱标签云和一个食材标签云。我们将

会把这些部分安置在它们各自的容器内，便于以后调整它们的位置。好了，让我们开始写 HTML 表单吧。

9.6.1 搜索表单

HTML 中的表单很简单，难的部分是表单和后台系统的交互。Foodbox 的搜索表单由两个元素构成：关键字字段和提交按钮。另外，一个比较难的部分是 HTML 表单需要将数据提交到 URL。创建表单的过程中，这个目标 URL 和服务器端处理数据的代码都是必不可少的。

还好，原有的 Foodbox 页面上已经有了这些信息，我们只要扫一眼就能发现下面的代码：

```
<form method="get" action="/recipes/">
  <input type="text" name="keywords">
  <input type="submit" value="search">
</form>
```

这段代码显示，提交表单的时候会用 GET 方法将数据传送到 `recipes`。同时我们还知道这个表单叫做 `keywords`。这差不多能让我们继续下去，只是有几个小问题需要解决。

首先 `input` 标签没有闭合。可能原来那些开发者忘记，或者他们用的 HTML 版本并不要求标签闭合，但是我们制定的 doctype 是需要的。当然，这个问题很好解决。

第二个问题是两个 `input` 标签都需要一个 `id` 属性，这会在添加样式的时候帮上大忙。最后，我们要把提交按钮换成一个放大镜的样子。

除了表单之外，搜索区域还需要一个标题，叫做 Search Result。

标题

在页面上添加文字很简单，只需要往指定区域打字就好了。但是你需要进一步考虑文字被呈现的效果。

在 HTML 中有好几种标签是用来标记文字的。实际上，光是跟标题有关的就有 6 个，分别是 `h1`、`h2`、`h3`、`h4`、`h5` 和 `h6`。其中数字越小，代表这个标签中的内容越重要。

一个网页至少要有一个用 `h1` 标记的大标题。搜索引擎会根据这些标题来确定页面在搜索结果中的权重。我们会在页面的主要区域用到大标题，其他区域则会用到 `h2` 标签。

在侧边栏中加入如下代码来构建搜索区域：

homepage_html/index.html

```
<div id="search">
  <h2 id="search_header">Search Recipes</h2>
```

```
    <form id="search_form" method="get" action="/recipes/">
      <div>
        <input type="text" id="search_keywords" name="keywords">
        <input type="image" alt="Search" src="images/search.png">
      </div>
    </form>
  </div>
```

分析一下上面的代码。侧边栏的搜索区域被独立包裹在自己的区域中，有自己的 id，这样以后添加样式的时候，你会有更多的灵活性。搜索表单重组了原来页面上的表单，新增了 id，同样这个 id 也是为了方便添加样式和 JavaScript 动作。

这里用一个 div 将输入区域装载起来。在 HTML 4.01 Strict 中，输入标签必须被放置在 div 或者其他块元素中，如标题标签或者段落标签。

最大的变化是，提交按钮不见了，我们用了一个图片按钮来代替它。

图片按钮跟普通的按钮作用一样，当用户单击它时，数据会被表单传送到相应的URL。区别在于，你可以用心仪的图片来替换原本只跟操作系统相关的无聊的默认按钮样式。[①]



小乔爱问……

我能够用一个链接来代替提交按钮放在表单之中吗

可以，但这不是个好点子。链接意味着获取信息，而按钮意味着发出消息。违背这个标准会造成一些不必要的可用性方面的麻烦。

如果要用链接来提交表单，就不得不用到 JavaScript——需要让链接去调用一个 JavaScript 来提交表单。我不推荐这种做法，它不是个好点子，所以我也不会教你怎么做。最直接的原因是，这样做会让那些不支持 JavaScript 的客户端无能为力。

当然，总有些人会声称这样做是合理的，他们的原因总离不开某种视觉效果。但是，图片状的按钮已经大大增强了表单的美观了。当然你也可以利用 CSS 来让按钮看起来像链接。

9.6.2 菜谱标签云

通常情况下，标签云的功能是在服务器端实现的，在那里会有某种机制从数据库里查询出最常用的标签。获取这些标签后，用 HTML 代码在页面上显示，然后通常用 CSS 来控制标签的样

① 表单中的很多东西，如单选或多选框、下拉菜单和按钮的样式都是由操作系统决定的，而且是没有办法自定义的。所以设计网站的时候，一定要在不同的操作系统上测试你的设计。

式，使它们的样式跟各自的流行程度相匹配。因为这是一本讲设计的书，所以那些服务器端的实现过程就不说了，我们还是专注于标签云的实现吧，这样你可以学到怎么为它们加上样式。

我已经说过，云中标签的样式跟它们的流行程度有关。如果有很多菜谱都被打上了某一个标签，那么可能需要我们给它一个大一点的样式；那些受欢迎程度差一点的标签的大小就相对小一些。为了简单起见，我们的标签云里有 5 个级别标签。其中使用次数最多的标签用级别 1，最少的那些标签用级别 5。

每个标签都是一个链接，链接到显示所有拥有这个标签的菜谱的页面。但是如何为这些标签加上样式呢？因为我们需要复用标签们的样式，所以这是个运用 `class` 属性的好机会。跟 `id` 属性一样，`class` 属性也是 HTML 文档中所有元素都可以有的。

链接是用 `a` 标签（`anchor` 标签）来定义的。你可以链接到同一个服务器上的其他文档，或者其他服务器上的其他文档，甚至是本页面的某一个部位，只要把 `a` 标签中的 `href` 属性值设置为你要链接到的 URL 就可以了。在开启和闭合标签之间的文字，称为超链接。我们来看看几种不同的超链接。

1. 绝对超链接

绝对超链接包含了链接源的完整地址，包括协议、服务器名和链接源在服务器上的地址：

```
<a href="http://www.google.com/">Google</a>
```

2. 相对链接

一个相对于当前路径的链接。你可以用相对链接定位到本目录下的子文件夹中的一个文件：

```
<a href="about/index.html">About Us</a>
```

也可以定位到本目录的父目录中的一个文件：

```
<a href="../index.html">Back to the home page</a>
```

还可以定位到跟网站根目录下的文件：

```
<a href="/index.html">Back to the home page</a>
```

你猜对了，这很像 Linux 文件系统上的文件浏览。

3. 锚点

你也可以链接到本页面上的某个位置，但是先要在页面上定义一个有名字的锚点：

```
<a name="ingredients"></a>
<h1>Ingredients</h1>
....
]]>
```

然后你就可以创建一个能够跳转到这个部分的链接：

```
<a href="#ingredients">Ingredients</a>
    ]]>
```

你也可以将锚点附在任何绝对或者相对 URL 的后面，把你的用户带到某个页面的某个位置：

```
<a href="http://www.yourfoodbox.com/recipes/55#ingredients">Ingredients</a>
```

锚点在那些内容丰富的长页面上极为有用，你可以用锚点来为这个页面建一个目录（table of contents，TOC），让用户可以跳到他们想要去看的内容。同样，你还能在每一部分结束的时候放置一个回到目录的链接，这样用户就不用自己滚动页面而回到目录了。

因为现在我们还处于制作演示页的阶段，虽然你可以把每个云里的标签都加上 URL，但是这样显然太浪费时间了，而且到最后你还是要用程序来生成这些链接的。所以就目前来讲，你可以制作单击后不跳转到任何地方的链接——只需要在需要链接的文件或地址属性后面写上“#”就可以了；这也是检查被点过的链接的好方法。你可以将这个技巧当做“链接灭火”运动。

现在可以开始创建侧边栏中的第一个标签云了：

homepage_html/index.html

```
<div id="browse_recipes">
  <h2 id="browse_recipes_header">Browse Recipes</h2>
  <a class="level_1" href="#">desserts</a>
  <a class="level_4" href="#">appetizers</a>
  <a class="level_5" href="#">indian</a>
  <a class="level_2" href="#">beef</a>
  <a class="level_5" href="#">entrees</a>
  <a class="level_4" href="#">mexican</a>
  <a class="level_3" href="#">seafood</a>
  <a class="level_4" href="#">drinks</a>
  <a class="level_2" href="#">pasta</a>
  <a class="level_1" href="#">italian</a>
  <a class="level_2" href="#">chicken</a>
  <a class="level_4" href="#">pork</a>
</div> <!-- end browse_recipes -->
```

每个超链接都被赋值了一个 `class` 属性，在添加样式的时候，每一种 `class` 都会被指定一种字号。跟在制作搜索区域的时候一样，我们也用一个 `div` 包裹这个区域，用 `h2` 标签呈现了这个区域的标题。

9.6.3 食材标签云

这个标签云跟 9.6 节中的菜谱标签云类似。只不过需要改动一些 `id`、标题和标签内容。从已经写好的 HTML 复制代码不是一件坏事。现在你不是在写程序，而是在标记内容，通常说的“重

复自己是种错误”这个规则不适用，你需要的是尽快完成工作。用正确的语义化结构标记内容不是什么很激动人心的工作。人们关心的只是最后的外观，而不是内在的运行原理。

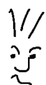
完成的新标签云看起来应该是这样：

homepage_html/index.html

```
<div id="popular_ingredients">
  <h2 id="popular_ingredients_header">Popular Ingredients</h2>
  <a class="level_2" href="#">oregano</a>
  <a class="level_4" href="#">garlic</a>
  <a class="level_3" href="#">black beans</a>
  <a class="level_3" href="#">apples</a>
  <a class="level_3" href="#">bananas</a>
  <a class="level_5" href="#">cheese</a>
  <a class="level_3" href="#">lettuce</a>
  <a class="level_1" href="#">chicken</a>
</div> <!-- end popular_ingredients -->
```

跟菜谱标签云一样，这个标签云的区域内也有自己的标题和链接。

好了，侧边栏做好了，你可以保存文件，然后在浏览器里看看效果。



小乔爱问……

井号到底是用来干什么的

“#”代表的是 HTML 文档中的位置，这样就可以创建跳转到页面内某位置的链接了。例如<ahref="index.html#news">News会载入 index 页面，并跳到页面内用锚点News标记的位置。

现在在标签云中，我们在连接的 URL 只用了一个“#”。浏览器会把这个链接理解为“跳到页面顶端”，基本上等同于什么也不做。

需要注意的是，你时常还会看到“#”作为一个占位符，放在链接到 JavaScript 的 a 标签中——这种标签的 onclick 属性里写的是 JavaScrip：

```
<a href="#" onclick="showAddUserForm(); return false;">
    Add New User</a>
```

这是一个很流行的解决方式，但是你应该尽可能避免使用它。那些无法使用 JavaScript 的用户单击这个链接后，会真的跳到页面顶端。所以你应该在这里用一个真正的链接。在上面那个“add user”的例子里，链接应该跳转到一个单独的添加用户页面。然后再用 Unobtrusive JavaScript 为单击事件添加动作。

9.7　主要内容

我们的主要区域里有以下几个部分：一大幅水平的图片、一列文字、注册和登录按钮，还有一个部分用来展示最新的菜谱。其中的三个部分需要用到我们做的样式页里的图片。我们还没有导出那些图片，所以可以先放一放，就像我们在做搜索框边上的图片按钮时那样。

9.7.1　意大利面图片

跟做 Banner 类似，我们要向 `img` 标签里加一个链接。然后把这个标签放在 `id` 是<main>的那对标签内：

homepage_html/index.html

```
<img id="main_image" src="images/pasta.jpg"
     alt="Pasta and marinara sauce">
```

我们还是假设有个叫做 images 的文件夹，就在这个 `html` 文件所处的目录下，并且 images 文件夹中有一张叫做 pasta.jpg 的图片。现在，浏览器会显示 `alt` 属性中的文字。

1. 默认文字

为了防止图片因为某种原因没有被加载，比如说用户用了屏幕阅读器或者文字版的浏览器，图片元素需要有一段默认文字。`alt` 属性就是用来添加这些文字的，使用这个属性的时候，要注意添加的文字应该具有描述性。虽然不设置默认文字，你也可以通过验证，但是这对你的用户没好处。记住，要用它来描述图片。

2. 文字内容

文字内容指的是 Get Cookin’s 的标题和那一段文字。这里的标题在写 CSS 的时候被图像所取代（其他标题也是一样）。<p>标签是用来包裹文字内容的，就像我们在搜索区域的标签云区域里做过的一样，标题和段落都由自己的 `div` 包裹着：

homepage_html/index.html

```
<div id="main_text">

  <h1 id="get_cooking">Get Cookin...</h1>
  <p>Foodbox is the best way to collect and share recipes
        with the rest of the world. You can build your own
        recipe book from thousands of great recipes from
        renowned chefs or users just like you.  You can also
        share your own secret recipes with a few of your friends
        or make them available to the rest of the world!</p>
```

```
  <p>Create an account today and get cookin!</p>
</div><!-- end main_text -->
```

注意，上面文字是被分成两段的。因为在样式页中，Create an account…那一段是跟前面一段分开的。也许你有过用
标签来换行的想法，但是你需要考虑一下内容的上下文。在这个例子里，两块文字是独立的两段话，所以你应该照如上方法将它们标记出来。



小乔爱问……

为什么这里的按钮用的是嵌入式，而不是用 CSS 给按钮加上图片，就像我们针对标题所计划的那样

在标题使用图片是为了保留字体的样式，而登录和注册按钮不仅仅是一段文字，它们还是界面控件的一部分。当然你也可以用 CSS 来做这个事情，参照第 12 章中的方法。

9.7.2 注册和登录按钮

我们会用图片来制作注册和登录按钮。这一块地方也应该被看做页面上的一个区域，你应该用一个有合适 id 的 div 标签把它包裹起来。

你希望这个按钮是可以单击的，但是不会提交任何数据，所以你不需要用表单按钮，只用一个带着超链接的图片就可以了。你还记得 a 标签吧？任何在开始和闭合标签之间的内容都会成为一个超链接，连图片也不例外，你会得到一个可以单击的图片。现在，在页面上插入两个 img 标签，分别用两个超链接标签将它们围起来，一个是注册按钮，一个是登录按钮：

homepage_html/index.html

```
<div id="signup_login">
  <a href="/signup/">
        <img src="images/btn_signup.png" alt="Sign up">
      </a>
  <a href="/login/">
        <img src="images/btn_login.png" alt="Log in">
      </a>
</div><!-- end signup_login -->
```

每个图片标签都有个 alt 属性，里面的文字跟按钮图片上的文字一样（这也是为了盲人和屏幕阅读器用户准备的）。这样添加上的图片周围会有一圈边框，用来表示这个图片是可以单击的。边框其实很不好看，我们稍后会用 CSS 将之去掉。

9.7.3　最新菜谱区

跟我们在制作标签云一样，你需要想出一个假菜谱放在这个区域里。最终，这部分菜谱是用程序从数据库查询出来然后放在页面上的。现在我们还不需要做出这个功能，创建这个页面的目的只是得到客户的反馈。

比起其他的，这个区域的样式稍微有些复杂。标题是个普通的标题，但是用来表述菜谱的段落有一点缩进。为方便起见我们会给段落标签添加一个 `class` 属性，因为 `class` 属性很适合给一组相同样式的元素（或者它们的子元素）使用。最终我们在添加样式的时候，就可以指定 `id` 为 `latest_recipes` 的 `div` 包围着的段落元素缩进显示。

homepage_html/index.html

```
<div id="latest_recipes">

  <h2 id="latest_recipes_header">Latest Recipes</h2>

  <div id="latest_recipe_1" class="latest_recipe">
    <h3><a href="#">Stuffed Chicken Breast</a></h3>
    <p>A lightly breaded breast of chicken stuffed with mushrooms
           and Swiss cheese. Easy to make even for beginners.</p>
  </div>

  <div id="latest_recipe_2" class="latest_recipe">
    <h3><a href="#">Chocolate Pancakes</a></h3>
    <p>This complete-from-scratch classic pancakes recipe is sure
        to please even the pickiest eater, especially chocolate
        lovers.</p>
  </div>

</div>
```

每个菜谱都有一个值为 `latest_recipe` 的 `class` 属性。跟 `id` 属性不同的是，`class` 属性可以被赋给不同的元素。在编写主要内容的 HTML 代码的时候，你就应该开始考虑如何用样式表实现设计效果之类的事情了。

这时如果 Photoshop 里的样式页在手边的话会很有帮助，因为这样你可以随时查看那些地方是共用同一种样式的。

9.8　页脚

页脚区域有页面的版权信息，还有隐私政策和使用条款的链接。在原来的那个网站上，版权信息用的是一个特殊字符，而且在一些浏览器中不能被正确显示。当开发者用 Dreamweaver、

Frontpage 这种视觉化编辑软件进行开发，然后从微软的 Word 程序中复制东西的时候，这类字符很容易在网页上出现。

像版权符号、特殊引号等其他各种特殊符号，都需要用实体代码（entity code）添加到网页上。

在页脚区域加上如下代码：

homepage_html/index.html

```
<div id="footer"> <!-- start of the footer -->

  <p id="copyright">Copyright &copy; 2010 Foodbox,
                      LLC, all rights reserved.</p>
  <p id="privacy_and_terms">
    <a href="terms.html">Terms of Service</a> |
    <a href="privacy.html">Privacy Policy</a>
  </p>
</div> <!-- end of the footer -->
```

`©`就是一个实体代码。浏览器碰到这样的实体代码时就会呈现出相应的字符。这样可以保证在用户不同的浏览器和字符集中，版权符号都会被正确的呈现。

在这个例子里，我为段落打上了带有唯一 `id` 的标签，而没有为它定义一个 `div`，因为 `p` 标签本来就是一个块元素，它之中的文本会独占一行，所以没有必要再加上一个 `div` 标签。

为了完成建立一个“内容灵活”的文档，你不需要在网页面上加太多的元素。

实体代码

你已经看到怎么用实体代码添加版权符号了。除此之外你还可以用它来做别的事情。

浏览器会忽略一个以上的连续空格，但是有时在段落中你会需要几个额外的空格。这时你可以用` `（non-breaking blank space，无打断空格的缩写）来强制浏览器显示空格。

你身边那些熟悉印刷品的人，像 PR 部门，可能不喜欢 HTML 中默认的引号，会要求使用弧形引号（curly quote）。那么你可以用“和”。

你还可以搜索一下 HTML 实体代码，会有很多例子。甚至像外文中令人厌烦的重读符号都有。

现在，你已经完成了整个页面，它看起来是这样的：

homepage_html/index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">

<html lang="en">

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Foodbox</title>
  </head>

  <body>
    <div id="page"> <!-- start of the page wrapper -->


      <div id="header"> <!-- start of header -->

        <img src="images/banner.png" alt="Foodbox">
      </div> <!-- end of header -->
      <div id="middle"> <!-- container for the sidebar and main region -->
        <div id="sidebar"> <!-- the sidebar -->
  <div id="search">
    <h2 id="search_header">Search Recipes</h2>
    <form id="search_form" method="get" action="/recipes/">
      <div>
        <input type="text" id="search_keywords" name="keywords">
        <input type="image" alt="Search" src="images/search.png">
      </div>
    </form>
  </div>

  <div id="browse_recipes">
    <h2 id="browse_recipes_header">Browse Recipes</h2>
    <a class="level_1" href="#">desserts</a>
    <a class="level_4" href="#">appetizers</a>
    <a class="level_5" href="#">indian</a>
    <a class="level_2" href="#">beef</a>
    <a class="level_5" href="#">entrees</a>
    <a class="level_4" href="#">mexican</a>
    <a class="level_3" href="#">seafood</a>
    <a class="level_4" href="#">drinks</a>
    <a class="level_2" href="#">pasta</a>
    <a class="level_1" href="#">italian</a>
    <a class="level_2" href="#">chicken</a>
    <a class="level_4" href="#">pork</a>
  </div> <!-- end browse_recipes -->

  <div id="popular_ingredients">
```

```
    <h2 id="popular_ingredients_header">Popular Ingredients</h2>
    <a class="level_2" href="#">oregano</a>
    <a class="level_4" href="#">garlic</a>
    <a class="level_3" href="#">black beans</a>
    <a class="level_3" href="#">apples</a>
    <a class="level_3" href="#">bananas</a>
    <a class="level_5" href="#">cheese</a>
    <a class="level_3" href="#">lettuce</a>
    <a class="level_1" href="#">chicken</a>
  </div> <!-- end popular_ingredients -->
</div> <!-- end of the sidebar -->

<div id="main"> <!-- start of main content -->

  <img id="main_image" src="images/pasta.jpg"
      alt="Pasta and marinara sauce">

  <div id="main_text">

    <h1 id="get_cooking">Get Cookin...</h1>
    <p>Foodbox is the best way to collect and share recipes
          with the rest of the world. You can build your own
          recipe book from thousands of great recipes from
          renowned chefs or users just like you.  You can also
                  share your own secret recipes with a few of your friends
                  or make them available to the rest of the world!</p>

            <p>Create an account today and get cookin!</p>
          </div><!-- end main_text -->
          <div id="signup_login">
            <a href="/signup/">
                  <img src="images/btn_signup.png" alt="Sign up">
                </a>
            <a href="/login/">
                  <img src="images/btn_login.png" alt="Log in">
                </a>
          </div><!-- end signup_login -->
          <div id="latest_recipes">

            <h2 id="latest_recipes_header">Latest Recipes</h2>

            <div id="latest_recipe_1" class="latest_recipe">
              <h3><a href="#">Stuffed Chicken Breast</a></h3>
              <p>A lightly breaded breast of chicken stuffed with mushrooms
                    and Swiss cheese. Easy to make even for beginners.</p>
            </div>

            <div id="latest_recipe_2" class="latest_recipe">
              <h3><a href="#">Chocolate Pancakes</a></h3>
              <p>This complete-from-scratch classic pancakes recipe is sure
```

```
                    to please even the pickiest eater, especially chocolate
                    lovers.</p>
              </div>

            </div>

          </div> <!-- end of main content -->
        </div> <!-- end of middle container -->
        <div id="footer"> <!-- start of the footer -->

          <p id="copyright">Copyright &copy; 2010 Foodbox,
                                LLC, all rights reserved.</p>
          <p id="privacy_and_terms">
            <a href="terms.html">Terms of Service</a> |
            <a href="privacy.html">Privacy Policy</a>
          </p>
        </div> <!-- end of the footer -->
      </div> <!-- end of the page wrapper -->

    </body>
  </html>
```

9.9　验证标签

之所以让你手写 HTML，目的是为了得到一个合法的文档。制定 HTML、XHTML 和 CSS 标准的 W3C 组织提供了一个在线工具，可以让你检查任何页面。你可以提供页面的 URL 或者将页面源码复制粘贴到工具中来检测其合法性。

有些具有 HTML 编辑功能的文本编辑器提供了针对本地文件的验证功能，但是我更喜欢结合 Firefox 浏览器和 Web Developer 工具栏来做这项工作。这个组合可以跨平台使用，而且是一直可用的。

9.9.1　为网页开发设置Firefox浏览器

Firefox 是一款很受欢迎的浏览器，同时它也是一个很好的网页开发工具。你可以为它安装插件和扩展包来增强它的功能，所以我们会用 Firefox 和一些扩展包来帮助我们测试网站和网络应用。

如果你还没有安装最新的 Feirfox 浏览器，去他们的网站[①]下载一个装上，然后启动浏览器。

---

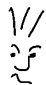
① http://www.getfirefox.com/。

9.9.2 Web Developer工具栏

Web Developer 工具栏可以将浏览器变成一个强大的网络开发环境，而且是专为网络应用开发者和网页设计师准备的。工具栏能让你轻松地用 W3C 的标准检测自己的页面，并且还自带一个实时的 CSS 编辑器，我们下一章将会用到它。[①]

在 Firefox 中输入 https://addons.mozilla.org/firefox/60/，进入 Web Developer 工具栏的下载页面，在页面上单击“Add to Firefox”链接，在弹出的对话框中单击安装按钮，安装该工具栏。

安装完这个扩展包之后，需要重启浏览器，重启之后，我们刚刚安装的工具栏就在收藏夹栏的下方。



小乔爱问……

为什么在多数人使用 IE 的时候，我们还要用 Firefox 开发

对比 IE，用 Firefox 开发可以节省大量的时间。这是因为 Firefox 在呈现页面的时候一丝不苟，不像 IE 会玩忽职守地放过一些错误，最终导致页面的样式表变成一个噩梦。当然，你还要在 IE 下测试网页，但是我们的策略是先在 Firefox 下开发，然后用一些专为 IE 准备的技巧改造页面，比如一些用注释写的条件语句为 IE 专门准备一些样式。按照这个策略，你能节省大量的时间。

随着几个新版的发布，IE 也慢慢地在遵守规范，而且在尽可能多的平台上用尽可能多的浏览器测试对于你来说也很重要。但是由于 Firefox 对网页标准的严格支持，另外再加上它强大而有帮助的各种插件，所以我推荐使用 Firefox 开发网页。

Linux 下的 Firefox

Linux 用户需要参考各版本的文档。你可以从源代码编译一份，但是很多版本的软件管理系统中就有现成的安装包可以用，像 Ubuntu 用户就可以用下面的命令安装：

```
sudo apt-get install mozilla-firefox
```

① 程序员可能会对工具栏的 session cookie 清除和查看报头的功能感兴趣。

Firebug

Firebug[①]这个工具让查看页面的 HTML、CSS 和 JavaScript 变得异常容易。虽然本书中我们不会使用它，但是你应该能体会到它是一个珍贵的工具。它可以让你查看所有 CSS 的定义、宽和高以及其他元素的属性。Firebug 基本上可以算是网络开发者的调试工具。

Firebug Lite[②]是它的跨浏览器版本，能帮你从 IE 的梦魇中解脱出来。

9.9.3 验证文档

验证不需要花太长时间。如果你的页面没有错误，将会有一条友好的信息告诉你页面是合法的。如果你看到了错误信息，那么验证报告会告诉你哪里出错了。如果这种情况发生了，你应该从页面的顶端开始排查修复。一个顶端的小错误可能会引起十个其他地方的错误。解决掉一个问题之后，再重新验证一次。

验证器甚至能检测出符号错误，比如&的使用错误。很多开发者在 URL 加入查询字符串发送到服务器：

http://www.example.com/search?first_name=homer&last_last_name=simpson

虽然这样的 URL 用起来没问题，但是验证器还是会将它列为不合法。标准的做法是将它用`&`替代。虽然在现代浏览器中很少会由&符号引起问题，但是偶尔蹦出的错误也时常发生。

9.10 HTML 5

在写这本书的时候，HTML 5 的仍然属于起草阶段，但是很多人已经开始做针对它的适应工作了。虽然还不是所有的浏览器都支持 HTML 5，但是它是向后兼容的。而事实上，HTML 5 的 doctype 也会强制 IE6 用标准模式呈现。这种兼容性可以让我们用 CSS 做出代有外观的网页。“HTML 5 展览室”[③]列出了已经开始像 HTML 5 靠拢的网站。

HTML 5 之所以如此吸引人，是因为它更加看重对内容的标注。在本章中，我们用 `div` 元素将标题、侧边栏、主内容和页脚标记出来，但是如果我们用 HTML 5，页面看起来会是这样的：

① http://getfirebug.com/。

② http://getfirebug.com/lite.html。

③ http://html5gallery.com/。

homepage_html/index_html5.html

```
<!DOCTYPE html>

<html lang="en-US">
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Foodbox</title>
  </head>

  <body>
    <section id="page">

      <header id="header">
        <img src="images/banner.png" alt="Foodbox">
      </header>
      <section id="middle">
        <aside id="sidebar">

          <section id="search">
            <h2 id="search_header">Search Recipes</h2>
            <form id="search_form" method="get" action="/recipes/">
              <div>
                <input type="text" id="search_keywords" name="keywords">
                <input type="image" alt="Search" src="images/search.png">
              </div>
            </form>
          </section>

          <section id="browse_recipes">
            <h2 id="browse_recipes_header">Browse Recipes</h2>
            <a class="level_1" href="#">desserts</a>
            <a class="level_4" href="#">appetizers</a>
            <a class="level_5" href="#">indian</a>
            <a class="level_2" href="#">beef</a>
            <a class="level_5" href="#">entrees</a>
            <a class="level_4" href="#">mexican</a>
            <a class="level_3" href="#">seafood</a>
            <a class="level_4" href="#">drinks</a>
            <a class="level_2" href="#">pasta</a>
            <a class="level_1" href="#">italian</a>
            <a class="level_2" href="#">chicken</a>
            <a class="level_4" href="#">pork</a>
          </section>

          <section id="popular_ingredients">
            <h2 id="popular_ingredients_header">Popular Ingredients</h2>
            <a class="level_2" href="#">oregano</a>
            <a class="level_4" href="#">garlic</a>
```

```
      <a class="level_3" href="#">black beans</a>
      <a class="level_3" href="#">apples</a>
      <a class="level_3" href="#">bananas</a>
      <a class="level_5" href="#">cheese</a>
      <a class="level_3" href="#">lettuce</a>
      <a class="level_1" href="#">chicken</a>
    </section>
  </aside>

  <section id="main">
    <img id="main_image" src="images/pasta.jpg"
         alt="Pasta and marinara sauce">

    <article id="main_text">
      <h1 id="get_cooking">Get Cookin...</h1>
      <p>Foodbox is the best way to collect and share recipes
             with the rest of the world. You can build your own
             recipe book from thousands of great recipes from
             renowned chefs or users just like you.  You can also
             share your own secret recipes with a few of your friends
             or make them available to the rest of the world!</p>

      <p>Create an account today and get cookin!</p>
    </article>

    <section id="signup_login">
      <a href="/signup/">
              <img src="images/btn_signup.png" alt="Sign up">
            </a>
      <a href="/login/">
              <img src="images/btn_login.png" alt="Log in">
            </a>
    </section>

    <section id="latest_recipes">
      <h2 id="latest_recipes_header">Latest Recipes</h2>
      <article id="latest_recipe_1" class="latest_recipe">
        <h3><a href="#">Stuffed Chicken Breast</a></h3>
        <p>A lightly breaded breast of chicken stuffed with mushrooms
               and Swiss cheese. Easy to make even for beginners.</p>
      </article>
       <article id="latest_recipe_2" class="latest_recipe">
         <h3><a href="#">Chocolate Pancakes</a></h3>
         <p>This complete-from-scratch classic pancakes recipe is sure
             to please even the pickiest eater, especially chocolate
             lovers.</p>
       </article>
     </section>
   </section>
 </section>
```

```
      <footer id="footer">
        <p id="copyright">Copyright &copy; 2010 Foodbox,
                            LLC, all rights reserved.</p>
        <p id="privacy_and_terms">
          <a href="terms.html">Terms of Service</a> |
          <a href="privacy.html">Privacy Policy</a>
        </p>
      </footer>
    </section>

  </body>
</html>
```

这些代码的描述性比 HTML 4.01 Strict 更强，但是由于 HTML5 仍属于不稳定阶段，我还是把注意力集中在 HTML 4.01 Strict 上。如果你想更超前，可以在接下来的练习里尝试用 HTML5 的模板。[①]

9.11 小结

在本章中，你学会了如何建立一个合法的页面，并且如何将页面内容结构化标记，以方便以后加上样式。你可以将这个结构用在以后的项目中，因为它只包含了少量的内容和结构化的元素。在本章中，你要掌握的主要内容是将文档结构化并且语义化地标记出来，同时保证代码灵活多变且符合规范。我们从逻辑上将相同的元素用标签分隔成组，然后用 HTML 的各种标签处理各组自己的内容。现在我们会用 CSS 将这个页面大变样。

① 一些老一点的浏览器可能不认识诸如 aside 一类的标签，所以无法加载它们的样式。但是你可以用 JavaScript 中的 document.createElement()方法来强制浏览器认识这个标签，只不过这需要用户的浏览器激活 JavaScript 功能。

第 10 章

为样式页面添砖加瓦

我们已经垒起了页面框架，现在要做的事情是把 Logo 和其他的图片准备好，让它们适合网络发布。并且在最终版本的网页中，我们也会用到这些图片。在本章中，你会了解到适合在网页上使用的各种图像格式，以及如何从 Photoshop 文档中将某片区域导出成独立文件。然后，你就可以在 HTML 中使用这些图片了。

10.1 图像优化

在导出图像之前，先要熟悉几个关于网络图像的概念，比如说文件大小、文件类型和图片优化。实际上，市面上有很多软件能帮你做这些优化工作，但是为了能让优化出的效果更符合需求，你还是应该了解为什么要做优化，如何做优化。

所谓图像优化，指的是在缩减图片文件体积以便网页使用的同时，保持图像的质量和清晰度。这样做可以带来如下益处。

- **对于用户来讲，小体积的图片更友好。**对图像进行优化之后，人们下载网页的速度会提高，这就意味着网站速度变快了；反之，如果页面上全是巨大的没有优化过的图像，因而导致人们要等半分钟才能看到内容，用户一定会不耐烦的。
- **小体积图片可以省流量。**网站主机通常会限制网站每个月的可用流量。如果网页中的图片体积小一些，额定流量就能晚些用完。有些主机提供商会在超过额定流量之后收取额外的费用。就算是自己准备主机的商业网站，也需要为流量付费。这个省钱的方法听上去不算什么，但是日积月累起来也是很可观的，特别是当网站的独立访问量达到一定规模的时候。
- **小体积图片可以省空间。**的确，现在大家都在说硬盘便宜了，不过如果网站不用那种巨大体积的图片，那在存储方面还是能省下很多钱的。亚马逊的 S3 服务费用既包括流量费用又包括存储空间费用，你应该尽可能将图片的体积缩小以节省存储费用和流量费用。

下载次数

虽然现在高速互联网已经比较普及了，但还是应该考虑页面资源的下载次数问题。一个 100 KB 的 JPEG 图片可能不怎么大，但是如果有 5 张这样的图片，外加 122 KB 的 JavaScript 原型库和几个 CSS 文件及其他内容，下载这些图片的时间就可能达到好几秒甚至更多。用户通常没多少耐心，所以你需要让自己的网页尽快显示出来。

有很多方法可以让你算出一个页面的大小。你可以手动计算网页、脚本、CSS 和图片文件的大小和。更简单的方式是用第三方的服务来帮你计算。在网页 http://www.websiteoptimization.com/services/analyze/上，输入一个网址，就能看到一份详细的报告。[①]

10.2　处理不同格式的图像

图片的类型有很多，优化的工作并不简单。比如说，一张照片和一个 Logo，它们所需要的优化是不一样的。

在浏览器中，你会面对三种主要的图像文件格式：GIF、PNG 和 JPEG。这三种格式的图像我们都会讲到。

10.2.1　GIF

GIF 是 Graphics Interchange Format 的简称。这是一种拥有 256 种 24 位 RGB 颜色的图片格式。虽然受限的颜色种类让它不适合做照片，但是用来做 Logo 图片还是很不错的，另外它还支持动画。

通常 GIF 文件都被用来作 Logo 或者按钮，因为它支持透明背景——这意味着你可以看到部分图片后面的背景。但是大多数开发者都转向 PNG 的怀抱了，因为 PNG 对透明的支持更好。

GIF文件优化

GIF 图像最多只能有 256 种颜色，优化图像就需要减少文件中所存储颜色的种类。如果 Logo 中只有 16 种颜色，那么你就将软件设置为只输出 16 种颜色。减少颜色就缩小了文件大小，但是图片可能变得惨不忍睹。越复杂的图片需要越多的颜色，Photoshop 提供了预览功能，这样你就能看到图片在优化设置下所呈现出的状态。

减少 GIF 图片的颜色数，要按照 2 的次方数来减少：16、32、64、128 和 256 都是合理的 GIF

---

① 想要用这种方法，网页必须已经处于向外界发布的状态。如果还没有到那个状态，你可以人工计算大小，或者利用 Dreamweaver 等类似的工具，这些工具能为你生成一份关于网页大小和预估下载时间的详细报告。

颜色数。如果你把这个数值设为非 2 的次方，文件的体积不会有任何变化；如果这个数小于 16，那么图像会出问题，可能根本就不能呈现出来，所以你不用考虑 16 种以下的颜色。

Photoshop 的 Save for Web & Devices（存储为 Web 或设备所用格式）功能提供了控制 GIF 颜色数量的功能（参见图 10-1）。

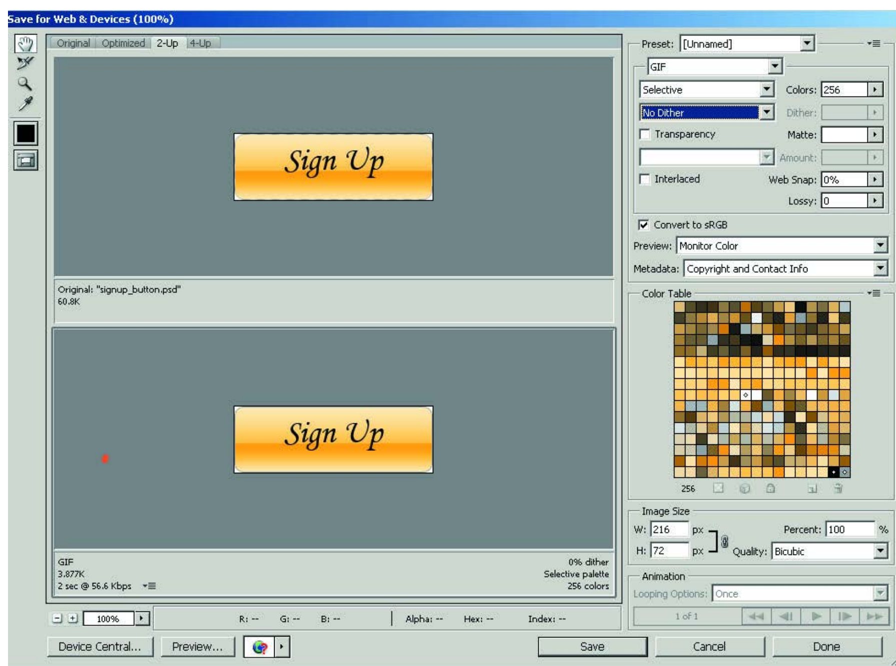


图 10-1　Photoshop 的 Save for Web & Devices 选项中有若干个设置，能为 GIF 图像自动挑选所需的颜色

10.2.2　PNG

PNG 是 Portable Network Graphics 的简写，是 GIF 的替代品。PNG 是一种采用无损压缩的位图图像格式。它只支持 RGB 颜色空间，专为网络使用而设计，同时对透明支持得非常好。虽然这种格式的透明效果只有较新的浏览器才支持，但是不要因此就放弃它，因为这种格式的图片质量非常高。

PNG文件优化

优化 PNG 图像需要做的是，为图像选择不同的“位深度”（bit depth）。图像越复杂，文件的体积越大。但是 PNG 跟 JPEG 的不同之处在于，PNG 采用的是无损压缩的方式。所以 PNG 很适合 Logo 和其他非照片图像，像图标、带有底纹、阴影或者发光效果的按钮等。

在使用 PNG 文件的时候要特别注意文件大小，一个 24 位带有透明效果的 PNG 文件可能会非常大。

10.2.3 JPEG

JPEG 是一种适合照片、图片的压缩格式，它被广泛支持，采用有损压缩。所以如果你过度压缩或者重复压缩了 JPEG 文件，图片上会留下痕迹。

因为 JPEG 并不支持透明，所以它只适合照片图片。像 Logo、截屏和渐变图形等肯定是不适合用 JPEG 文件的。

不要重复压缩 JPEG 文件

千万不要重复压缩 JPEG 文件。如果原始文件用 20%的压缩率被压缩了，效果仍然不满意，不要用新得到的文件再进行压缩，而是应该重新压缩原始文件。要记得保留一份原始的未压缩文件，这样就能一直使用这个原始文件进行压缩了。

JPEG文件优化

优化 JPEG 文件很简单，压缩图像让它的体积变小就可以了。压缩虽然可以有效地压缩文件体积，但也会为图片带来质量损失。你需要在文件大小和图像质量之间权衡利弊。

你可以在保存 JPEP 文件的时候压缩 JPEG 文件。在 Photoshop 中，当你选择将文件保存为 JPEG 格式时，就会有选择压缩等级的选项（参见图 10-2）。所有的图像处理软件在压缩 JPEG 的时候都是一样的，它们会让你选择压缩等级，这时压缩等级就代表了图像质量。更大的压缩比代表了更差的图像质量。

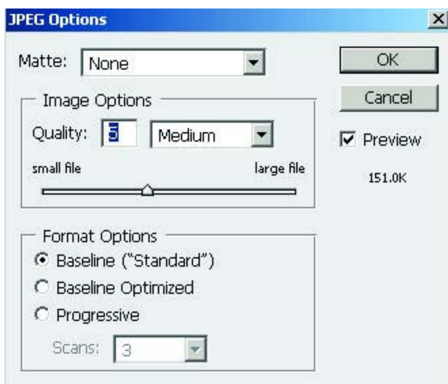


图 10-2 压缩 JPEG 文件

如果在压缩之后图像文件的体积还是很大，那么就只剩下最后一个方法了：减小图片的宽和高。

> **数码相机**
>
> 一些消费者导向的数码相机会把照片存为 JPEG 格式。如果你的相机就是这样的，我强烈建议你向厂商咨询一下，问问那些照片被压缩到什么程度了。如果可能，最好想办法得到未经压缩的图片。不少相机都有调整照片存储方式的选项。我建议用 RAW 格式存储照片，然后用 Photoshop 将它们转换为 JPEG 文件。如果需要更多信息，看一看你的相机说明书。
>
> 如果照片是从专业摄影师那里拿到的，记得向他们索要 RAW 文件或者 DNG（Digital Negative Files，数字负片文件），而且要记住，在摄影师拍照之前就要将这些要求告诉他。

10.3　文档切片

打开样式页文档，用 Slice（切片）和 Slice Select（切片选择）工具从这个文件中切出几张图片。只要我们做出了正确的设置，Photoshop 就会为我们自动完成图像优化。我们并不需要用软件把整个图片都自动转化为网页，只需要从中切割出几个图片给样式表用就好了。

样式页确认

要保证所有的元素都坐落在网格线内，因为这些网格线将是切片的辅助线，我们肯定不希望不小心将文字或者图片切开。将图片放大到 300%，然后按住 Space 键激活手形工具，按住 Space 键的同时按住鼠标，左右移动图片，确保 Logo、标题和图片什么的都恰好在网格线之中，文字或者图片的边缘都没有超过网格线。图 10-3 就提供了一个反例。



图 10-3　超过网格范围的元素

如果真的有什么东西越过了网格线，选择 Move 工具然后右键单击（Windows 系统）或者按住 Command 键的同时单击（Mac 系统）越过网格线的那部分，就会弹出一个菜单，显示鼠标下的所有图层的名字。选择那个出轨的元素，然后用方向键慢慢地将该元素移动到它应该存在的位置。

10.4 创建切片

我们会在这个文档中创建几个不同的切片，虽然它们将被存为不同格式的文件，但是切片的过程是一样的。我们先从 Foodbox 的 Logo 下手。

从工具栏选择切片工具[1]，用它在 Logo 周围画一个方框，方框的左上角应该与 18 像素位置的水平网格线和 72 像素位置的竖直网格线的交点重合，方框的右下角则应该位于 552 像素位置的竖直网格线和 108 像素的水平网格线的交点之上。记得勾选 Snap to Grid（对齐网格）选项，这个选项能加快对齐的操作。

从辅助线切片

如果你够勤快，在添加每个元素的时候都作了辅助线，那么你完全可以跳过这些手动切片的步骤。选中 Slice 工具的时候，你可以去点那个 Slices from Guides（从辅助线切割）的按钮。但是前提是辅助线一定是到位而且准确的，但这样可能会得到不少多余的切片。如果辅助线不到位，你可能会不小心把一张完整的图片切开。

不管你需要切多少切片，这种方法都会帮你节省一点时间——即便在切完之后你还需要做一些手工调整。

很多 GIMP 的用户会用这个方法来切图。

然后选用 Slice Select（切片选择）工具——这个工具应该位于 Slice 工具的下方，按住工具栏中的切片按钮保持不动，你就能看到它了。用该工具双击 Logo 切片。将切片的名字设置为 Banner，这里设置的名字就是该切片将要被保存的文件名字。要记得为每个文件命名，否则 Photoshop 会自动生成名字，从而让你的文件变得混乱。[2]

---

① 在 Photoshop CS4 之前的版本中，切片工具在工具栏中是一个单独的图片，但是在 Photoshop CS4 中，它位于 Crop（剪切）工具之后。

② 为切片命名的另外一个好处是帮助你区分所需要的切片和 Photoshop 自动创建的切片。切片的时候总会有些重叠的文件或者你不需要的文件，这时文件名就能帮你找出哪些是需要导出的切片。

将剩下的图片也做成切片

用同样的方法将其他图片也切割出来，同时记得为它们命名。你总共要切出以下图片：

- ❑ 搜索菜谱部分的顶部（search_recipes）
- ❑ 搜索按钮（search）
- ❑ 菜谱浏览区域的顶部（browser_recipes）
- ❑ 最受欢迎食材的顶部（popular_ingredients）
- ❑ 意大利面的图片（pasta）
- ❑ Get Cookin'标题（get_cookin)
- ❑ 最新菜谱的标题（latest_recipes）
- ❑ 登录按钮（btn_login）
- ❑ 注册按钮（btn_signup）

切片的时候要注意沿着网格线切割。如果有哪个图片越过了某条网格线，那么就将切片扩大到与之相邻的另外一条网格线上。切出来的图片应该正好是 18 像素的倍数，使其限定在基本网格线内。[①]在图 10-4 中，你可以看到完成切片的样式页。



图 10-4 样式页切片完成后的样子

① 虽然你可以切出任何大小的图片，但是你要做好用 CSS 为图片加上外边距（margin）和内边距（padding）的准备，好让图片仍然处于网格之中。

为了使之后的 CSS 的工作变得更简单，最好让侧边栏的标题的宽度和高度保持一致，分别为 180 像素和 36 像素。然后把 Get Cookin’和 Latest Recipes 这两张图的大小统一为长 198 像素，宽 54 像素。

设定切片名字的时候你可以检查它的大小。以 Get Cookin’切片为例，双击之后，信息对话框中的 X、Y 坐标代表的是切片的起始点，高和宽的数值则是相对于那个起始点坐标的距离。Get Cookin’切片的起始坐标应该是 X 为 378、Y 为 252，它的宽是 162 像素，高是 54 像素。

保存文档的同时，切片信息也被保存下来了，这样下次再打开的时候就不用再去切片了。

10.5　将 Banner 导出成 PNG 文件

虽然你可以将 Logo 导出成任意类型的文件，但是在这个页面上，我们会用 PNG 格式。PNG 不但是无损压缩，还支持显示很多颜色。而且页面上的 Logo 有渐变消失的反射效果，这个效果对于标准 GIF 来说太复杂了，而 JPEG 的压缩又会让 Logo 显示不正常。

Logo 图片位于一片黄色背景之上，所以我们没有必要将它弄成带有透明背景的图片，那样不会让人察觉到任何变化。但是我们找到了一个很好的借口，那就是借此机会让你学习如何制作透明的 PNG 图片，这样等你再遇到需要的情况时就能自己完成了。在 Photoshop 中，为了能导出半透明的图片，你需要将该图片下的所有图层都去掉。这意味着首先要把背景图层变成一个普通图层。在图层面板中找到背景图层，双击该图层，在弹出的 Layer Properties 对话框中将这个图层命名为 background_layer，然后按确认按钮。

10.5.1　隐藏图层

为了导出带透明背景的 PNG，你得把 Banner 后面的所有图层都隐藏起来。单击背景图层和黄色页头图层左侧的眼睛标志就可以隐藏图层。然后 Photoshop 会显示一个棋盘状的背景，你的 Banner 就会看上去和图 10-5 一样了。



图 10-5　Logo 背后的透明图层

10.5.2　保存切片

Photoshop 中的 Save for Web & Devices 菜单选项是专为保存网络优化图片设置的，普通的保

存按钮就没有这个功能。选中这个选项后，你会看到文档的预览图，里面的切片都已经就绪。

选择包含 Logo 的那个切片，这时右边的侧边栏会显示相应的切片属性。你可以为每个切片单独设置，可以把它们导出成 PNG、JPEG 或者 GIF 等不同的格式。

将导出类型设置为 PNG-24，注意 Transparency（透明）这个选项要勾选上。选中这个选项后，在预览中看到的图片背景就和你在画布上看到的是一样的了，都是棋盘状的。如图 10-6 所示。

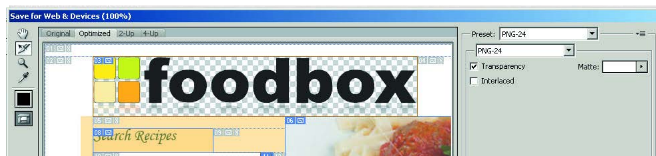


图 10-6 导出 Foodbox 的 Logo

单击保存按钮会弹出 Save Optimized As（优化保存为）对话框。将保存地址设置为 Foodbox 的项目文件夹中，Photoshop 会为你创建一个 images（图片）文件夹。文件名不用改，但是将保存类型改为 Images only（只保存图片），将 Slices option 改为 Selected Slices。文件名将和你之前设置切片属性时输入的一样。

保存后确认一下 Banner.png 是不是存到 images 文件中了，然后开始准备导出其他的图片。

10.6 导出其他图片

取消背景图层和页头图层的隐藏，然后再选择 Save for Web & Devices 选项。选中意大利面的图片，将类型改为 JPEG 图片，设置图片质量为 80。质量越好，图片文件体积越大。所以你要做好权衡。

选中登入按钮切片，格式设置成 PNG 8，注意不要勾选 Transparency 选项。注册按钮和搜索图标也一样处理。之所以不用 24 位的 PNG 格式，是因为这些按钮的颜色不多，8 位正好。

标题图像需要存为 GIF。[①]选中 Get Cookin’切片，按住 Shift 键，单击其他的标题，这样可以选中所有的标题切片，然后统一把它们的格式改为 GIF，这样的设置会对所有的切片生效。

继续按住 Shift 键，单击意大利图片、搜索按钮、登入按钮和注册按钮的切片。然后单击保存按钮，将所有的切片存进 images 文件夹。在 Save Optimized As 对话框中采用上述相同的设置，所有的图片都会导出到正确的位置。

① 这里完全是为了练习演示的需要才用 GIF 的，你完全可以用 PNG-8 来代替。

图片全部导出之后记得保存一下 Photoshop 文档，这个软件会把切片和它们各自的设置保存起来，以便于以后的调整和导出。

10.7 小结

本章里你学到了网页中不同图片文件的类型，并且知道了如何从设计好的样式页中对这些图片进行切片。利用 Photoshop 的切片功能来做图片优化对于日后网页的外观改进是很有帮助的，因为你只需要改动样式页然后重新导出图片就可以了。

接下来该装扮页面了。

第 11 章

使用 CSS 布局

为了将纸上的设计稿实现成网页，我们已经走过了一段很长的路，现在终点就在前方。马上，我们会接触到网页设计中最不好理解的概念：用 CSS 来安排各个元素的摆放位置。这个东西具体实现起来貌似不难，但是如果要真正理解它背后的原理，还是需要一番周折的，至少比决定配色方案要困难许多。这一章的任务，就是指导你了解 CSS 的一些原理和技巧，让你可以用它将没有样式的页面变得跟样式页几乎一模一样。

11.1 浏览器招人厌

如果不是浏览器捣乱（更准确地说，是浏览器制造商），基于 CSS 的网页设计会容易很多。然而，我们生活在一个开源支持者和商业软件开发者之间战争不断的世界，人们总在争论标准应该如何实现。在前几章我们讨论过这个问题，比如 IE 和 Firefox 呈现内容的方式不一样。让设计支持两个浏览器已经够困难了，但是别忘记还有苹果公司的 Safari，各种版本的 Opera 和 Google 的 Chrome 浏览器。每种浏览器都有各自不同的优缺点，作为网页程序员，你的任务就是了解它们各自的强项和弱点，然后将你那引人入胜的页面呈现给使用不同浏览器的用户。

一定的 CSS 技巧可以帮助你骗过浏览器的 CSS 解释器，通过这种方法，可以绕过一些由 CSS 引起的问题。但是要注意，这是个很危险的方法。因为有些技巧依赖的是浏览器的 bug，[①]而那些 bug 最终是会被修复的。当开发者习惯了 IE6 和它的怪异之处后，可以写出 CSS hack 来让他们的网页显示正确。IE6 已经 6 年没有更新了，于是当微软发布 IE7 的时候，很多人发现他们的页面需要一次大修。你一定不想重蹈他们的覆辙吧？所以你要写的页面，应该是在未来仍然可以被正确呈现的页面。

① 这种依赖浏览器 bug 的技巧被称做 CSS hack，下文也许会用 hack 这个单词来代表 CSS hack。——译者注

不要只是复制代码

不要照抄从网上找到的代码！虽然在互联网上，你能找到很多漂亮的 CSS 代码，但是其中的很大一部分是浏览器 hack，或者是基于其他你不熟悉的臆断编写的代码。我想，你肯定不希望自己开发的软件中有自己不熟悉的代码吧。在使用那些 CSS 之前，先搞清楚这些 CSS 的原理是什么，就像你在写程序时所做的一样。

11.2 CSS 基础

CSS 是 Cascading Style Sheet（层叠样式表）的简称，是用来表述 HTML 文档表现形式的语言。大多数人在涉足网络开发的时候，都是用 CSS 来改变页面上的文字外观的。

很快，这些人发现他们可以用一套规则来确定一个页面中所有段落的样式，而无需单独地为数百个页面中的每个段落都单独设置样式。

这个用途其实只是冰山一角。利用 CSS，你可以为文档添加色彩和图片，甚至改变整个页面的布局结构。本章中，你将学到如何构建 Foodbox 的布局，如何将之前设计稿中的样式应用到文档的不同区域中。

一条 CSS 规则主要由选择符（selector）和声明（declaration）组成，如图 11-1 所示。接下来我们会详细探索关于 CSS 的一切。

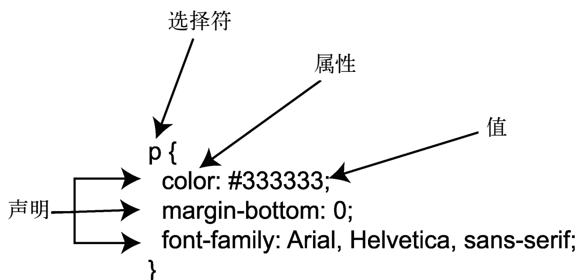


图 11-1 CSS 规则的组成部分

11.2.1 选择符

选择符规定了哪个或者哪些元素会应用某条 CSS 规则。选择符可以是 `html` 标签、`id` 或者 `class`。

不同类型的选择符

HTML 标签选择符就是把标签的尖括弧去掉。在下面的例子中，h1 标签的颜色会是蓝色的：

```
h1{
  color:#009;
}
```

标签 p、h1 和 body 都可以作为 HTML 标签选择符。

id 选择符总以井号（#）开头，用来指向 HTML 标签的 id 属性。下面的代码定义了 id 是 page 的元素的宽度。

```
#page{
  width:900px;
}
```

#page、#header 和#footer 都是指向 id 的选择符，而指向 class 的选择符则以英文句号（.）开头：

```
.box{
  border:1px solid #000;
}
```

.box、.important 和.newsitem 这几个选择符后面的规则会应用到相应的元素中。

11.2.2 声明：属性和值

声明定义了需要添加给相应选择符的样式，每一条声明都确定了一个 CSS 属性的值。如果想要将 h1 标签的文字颜色变成红色，只需要将颜色属性（color）设为#F00（红色的十六进制码简写）。下面的程序片段展示了用法：

```
h1{
  color:#F00;
}
```

每条声明中的属性和值都需要用冒号分开，而每一对属性和值则用分号隔开。[①]如果在一个 CSS 规则中有很多声明，那么你可以像这样把它们分开来写：

```
h1{
  font-size:24px;
  font-weight:bolder;
  color:#f00;
}
```

你也可以将所有东西写在一行：

① 一条 CSS 规则中最后的分号可以省略，但是不推荐这样做，因为之后你可能会在规则中添加新的声明，而忘记了那个省略的分号。

```
h1{font-size:24px;font-weight:bolder;color:#f00;}
```

这种集中在一行的写法很不利于阅读，但是没有了那些多余的空格，这种写法会省下不少带宽。可能在开发的时候，你有必要保持样式表的格式美观。在开发过程中你可以做一个工具用来在网站上线之前去掉那些空格和回车，这很容易。而且你也可以对 HTML 做同样的事情，因为它对空格和回车也不敏感。

11.2.3 关于“层叠”

没有必要在一条规则中将某个元素的样式全部定义一遍，你可以将这些声明分散在不同的文件中。另外，你还可以随时随地地添加内嵌的样式或者页面级别的样式：

```
/* Set the line height */
p{line-height:18px;}

/* set the color */
p{color:#003;}
```

这个特性让你能够以功能为区分标准，将样式表打散。你可以用一个 CSS 文件定义页面布局，另外一个文件专门定义字体和颜色。

然而，那些应用在统一元素上的不同规则之间可能会互相冲突。CSS 则有一套自己的优先级制度，我们将它称为*层叠*。一般样式表有三个来源：作者（也就是你）、用户（用户可以用自己的样式表取代你的）和浏览器默认值。层叠机制为每个 CSS 规则赋予了不同的权重。为了简单起见，我们这里只会涉及在你自己定义的样式表中发生冲突的情况，而不会涉及太多用户为浏览器做个性化设置而带来的不同效果。

你写的样式表中的规则权重要比浏览器或者用户的高，但是你写的这些样式的内部可能会出现矛盾。理解层叠的原理可以帮助你避免冲突，同时在必要时安全地覆盖某些样式规则。

1. id选择符

id 选择符所指定的元素很明确。比如说如果你将所有的 h2 标签都定义为蓝色字，但是希望其中的某一个是红色，那么你就可以为它单独设定一个 id，然后用相应的 id 选择符。

css_layout/examples/01_selectors.html

```
<h2>Products</h2>
<h2>Clearance Items</h2>
<h2 id="special_promotion">Hot Deals!</h2>
```

css_layout/examples/01_selectors.html

```
h2{color:#00F;}
#special_promotion{color:#F00;}
```

在这个例子里，Products 和 Clearance 会是蓝色的，而 Hot Deals！这个标题则会是红色的，就如图 11-2 里所显示的一样。通常情况下，id 选择器中的 CSS 规则会覆盖掉其他所有的规则定义。[①]

Products

Clearance Items

Hot Deals!

图 11-2 我们用 id 选择器覆盖了别的规则，从而让标题变成了蓝色

2. class选择器

class 选择器比一般的 HTML 标签选择器的优先级要高，但是比不上 id 选择器。我们为其中一个标题的 class 属性加上 promo 来看看效果：

css_layout/examples/02_selectors.html

```
<h2>Products</h2>
<h2 class="promo">Clearance Items</h2>
<h2 class="promo" id="special_promotion">Hot Deals!</h2>
```

然后再为这些标题加上样式：

css_layout/examples/02_selectors.html

```
h2{color:#00F;}
#special_promotion{color:#F00;}
.promo{color:#0F0;}
```

现在，Products 标题变成了蓝色，Clearance 是绿色，最后的 Hot Deals！还是红色，如图 11-3 所示。class 中的规则覆盖了 h2 里的规则，但是 id 选择器中的规则得以保留。

Products

Clearance Items

Hot Deals!

图 11-3 class 选择器的作用体现在第二个条目上，把它变成了绿色，但是它并没有覆盖到第三个条目上，因为 id 选择器里的规则不能被 class 选择器中的规则重载

① 其实用 class 会更好，因为 id 在一个页面上是唯一的。

3. 顺序很重要

如果没有优先级更高的规则，层叠机制会选用最新定义的样式加以应用。利用这条规则，你可以在每页或者每个元素上重写 CSS 样式，这也是为什么 CSS 如此好用的重要原因之一。通常多个页面共用一个样式表，而浏览器只要发现一个能用的样式，就会将它加载到元素上。了解其中的规律可以让你获得极大的灵活性。

我们用例子来说明。比如标题相关的 CSS 样式是这样的：

css_layout/examples/03_selectors_page_style.html

```
h2{color:#00F;}
#special_promotion{color:#F00;}
.promo{color:#0F0;}
```

然后你在页面上加入了如下的新样式:

css_layout/examples/03_selectors_page_style.html

```
#special_promotion{color:#FF0;}
```

在页面上的 CSS 规则的优先级要高于在样式表文件中设置的 CSS 规则，因为这相当于再次定义规则。在图 11-4 中，Hot Deals！ 这个标题显示的是黄色，而不是红色。

Products

Clearance Items

Hot Deals!

图 11-4 我们把 Hot Deals!这个标题的颜色定义了两次，浏览器会执行最新出现的那条声明

要记住 CSS 样式表文件整体不会受这种冲突的影响，只有其中个别的声明会波及。

在这一章中，我们会时常用到这种重定义的方法。但是大多数情况下，我们还是会尽可能地将相同的规则整合起来。我们的最终目的是将结构和样式分开，而并不是为了覆盖而覆盖。

4. !important的重要性

有些时候你需要改变 CSS 解析的规则来实现自己想要的效果。CSS 提供了`!important` 关键字让层叠机制执行一些规则，忽略掉本来的优先级顺序。在 11.2 节中讲到 `class` 选择器的时候，`class` promo 选择器中的规则不能改掉 `id` special_promotions 选择器中的规则，因为 `id` 选择器的优先级最高。但是，我们可以在 promo 规则的某些声明的后面加上`!important` 关键字，强制以最高优先级执行这条规则。

css_layout/examples/04_selectors_important.html

```
h2{color:#00F;}
#special_promotion{color:#F00;}
.promo{color:#0F0 !important;}
```

这下 Hot Deals!标题变成了绿色，如图 11-5 所示。

Products

Clearance Items

Hot Deals!

图 11-5 用!important 关键字来强制覆盖之前的规则

11.3 浏览器如何解析 CSS

你应该已经学会了如何定义样式，现在你需要了解的是，从浏览器的角度看，是怎样处理样式规则的。请求一个页面的时候，浏览器会开始解析 HTML 并呈现页面。如果在解析过程中遇到样式规则，或者是对样式表文件的引用，浏览器会在呈现页面的过程中在页面上应用这些规则。在页面上，一共有三种方式引用样式表。

11.3.1 嵌入式

所有的 HTML 标签都有一个 style 属性，在声明一个元素的时候，你可以利用这个属性来定义 CSS：

```
<h1 style="color:#f00;font-size:18px;">Welcome</h1>
```

开发者设计帮助功能或标签库的时候经常会用到这种方式。虽然这种方式看起来很方便，但实际上它是三种方法中最糟糕的一种，请想象以下的情况：

```
<h3>Services</h3>
<ul>
  <li style="color:#300;">Computer repair</li>
  <li style="color:#300;">Small business networking support</li>
  <li style="color:#300;">Computer hardware sales</li>
  <li style="color:#300;">Web development</li>
</ul>
```

你重复写了同一个声明，增加了HTML文档的代码量，同时还造成了样式信息无法重用的后果。如果这个列表会出现在多个页面上，那么你不得不将这条声明多次。而且，如果你的客户突然希望将颜色从红色改成蓝色，那你就需要改动无数的地方。而这种内容和设计混杂的情况，也是我们一直希望避免的。

但是这种写法也不是一无是处。有的时候，在一些特定的页面上，某些特定的元素需要微调，而你又不愿意把这个小小的改动放进全局共用的样式表中，此时这种写法就派上用场了。也就是说，不要滥用这个技术。后台程序员喜欢这样写：

```
<?php
  echo '<p style="font-size:18px;color=' . $color . ';">' . $description + '</p>';
?>
```

乍一看，这种写法是个不错的技巧，但是这种写法也让事后对颜色的修改变得异常困难。因为样式表改起来很简单，但是修改后台代码就很困难了。你最好别用这种方式写 CSS，这样不好。实际上，这就跟用 Comic Sans 字体一样糟糕。虽然这种写法看起来很有吸引力，但是你应该用一个 `class` 来代替，就像下面这样：

```
<?php
  echo '<p class="description">' . $description + '</p>';
?>
```

然后在文档的其他地方定义 `description` 类，改变文字外观，同时也不会影响后台代码。

11.3.2 style标签

HTML 中有一个 `style` 标签，能让你在文档的头部定义完整的样式表：

```
<style>
  body{
    font-family: Arial, Helvetica, sans-serif;
    font-size:12px;
    line-height: 18px;
  }

  h1{font-size:18px; line-height:36px;}
  h2{font-size:16px;}
  h3{font-size:14px;}

  #page{
    width:900px;
  }


</style>
```

如果有些页面需要自主独立的样式，这个方法显然是再适合不过了。但是问题也同样明显，跟上面那个嵌入式的写法一样，这个写法也是将内容和样式混杂在一起了，同时这些页面的样式也不能被其他页面共用。

当你头一次将某些模板写成 CSS 规则的时候，这种写法是很适合的，因为你不需要创建多余的文件。写完之后，这些在 `style` 标签内的 CSS 代码可以被剪切、粘贴到一个独立的 CSS 文件中，然后为其他页面所用。



小乔爱问……

那些三个数字的颜色代码是怎么回事？我以为颜色都是 6 个数字的呢

当每一对数字都由两个重复的数字组成时，CSS 提供了一种简写的方法。比如红色的颜色代码是#FF0000，这个代码表示：红色加满，绿色和蓝色一点都不加。这里因为 F、0 和 0 都重复出现了，所以你可以将它简写成#F00。简单说来，这又是一种减少文档中文字数量的方法。

11.3.3 外部CSS文件

`link` 标签可以让你在 HTML 文档中引入外部的样式表，就像引入外部 JavaScript 文件一样。网站用户的浏览器会下载这些文件，然后将样式加载到页面上。如果你有多个页面共用一个文件，用 `link` 引入外部文件可以大大增强用户的体验，因为你只需要将页面内容发送到用户端就好了，CSS 文件只需要传送一次就可以了。

大体上来说，这是使用样式表的最好方法，这也是在本章中会一直使用的方式。其他两种方法则适用于单独页面样式的偶尔微调。

11.4 创建并链接新的 CSS 样式表

打开文本编辑器，创建一个新的文档。将这个文档保存到 stylesheets 文件夹里，取名为 layout.css。在这个文件里，会定义页面上所有跟布局和对齐相关的东西。之后，会把字体和颜色的相关规则放在另外一个文件里面。

然后关闭那个文件。这次不会用文本编辑器去写 CSS，而是使用 Firefox 浏览器中的 Web Developer 工具栏。

打开首页 HTML 文件，在<head>…</head>区域内添加如下代码：

```
<link rel="stylesheet" href="stylesheets/layout.css"
      type="text/css" media="screen" charset="utf-8" />
```

上面的代码是一个链接外部样式表的示例，里面用到了一个相对链接指向 stylesheets 文件夹中的 layout.css 文件。跟 `img` 标签类似，`link` 标签中的链接可以是相对于根目录“/”的相对链接，也可以是一个连接到服务器其他位置的绝对链接。

样式表可以指定显示方式。举个例子来说，你可以指定某个样式表只在页面显示于屏幕之上的时候应用，也可以指定它在网页打印的时候应用。这为设计打印版本或者移动版本的网页提供

了便利。换句话说，就是不要太依赖于 media 这个属性。通常浏览器负责解析这些东西。但是有些浏览器并不会这样做，比如屏幕阅读器或某些移动设备上的浏览器。因此你一定要经常做测试。

11.5 定义基本结构、页头和页脚

在 Firefox 浏览器中打开 index.html 文件。此时，整个页面没有任何样式，但是仍然是可用并且可读的。这就是页面在一个不支持样式表或者是基于文字的浏览器中打开时的显示形态。现在让我们利用 Web Developer 工具栏为它带来一些改变，因为在 Web Developer 中做出的改动可以实时地显示出来。

同时按住 Ctrl+Shift+E，或者从菜单 CSS 中选择 Edit CSS（编辑 CSS），启动 CSS 编辑器。编辑器面板通常在浏览器底部出现。单击面板中 Position（位置）按钮，将编辑器栏拖放到文档的左侧。Position 按钮在 Edit CSS 标签的右侧。

无论是在页面的 head 标签内，还是在独立的样式表文件中添加的 CSS 代码都会出现在编辑器中，然后就可以修改它们。但是现在，我们面对的是一个空白的画布。

11.5.1 浏览器默认

每个浏览器都会以各自的方式显示页面：或者是不同的外边距（margin），或者是不同的行间距、字体大小，甚至还会有颜色的不同。这些对定义行高和其他元素造成了困难。但是通过在 CSS 中将一切清零，可以绕过这些默认值带来的麻烦，至少这种方法适合大部分主流的浏览器。将下面的规则添加到 CSS 编辑器中，可以看到行与行之间的间隙消失了：

css_layout/layout.css

```
body, p, h1, h2, h3, h4, h5, h6, ul, li, form{
  margin:0;
  padding:0;
  line-height:18px;
}

p, h2, h3, h4, h5, h6{
  margin-bottom:18px;
}
```

上面第一条规则移去了页面所有元素周围的外边距（margin，指的是元素周围的留白）和内边距（padding，指的是元素内部的留白）。同时，还将行距设置成 18 像素，重载了浏览器默认的行高值。

第二条规则将段落和标题的底部外边距设置为 18 像素，这个规则帮助我们将所有的元素都和网格对齐。

经常保存

你应该懂得经常保存文件的道理，但是在 Firefox 浏览器的 CSS 编辑器中，保存文件这个动作需要更加频繁。跳转到其他页面或者是刷新本页面都有可能将 CSS 文件重置，导致你的改动全部丢失。虽然实时显示改动是一件很好的事情，但是可能你中意的传统文字编辑器会让你感觉更舒服些。

共享规则

选择符可以放在一个小组里，这个小组里的所有选择符都会共用一套规则。虽然这一步并不是必要的，但这是一种减少文档中代码量的好方法。参看以下规则：

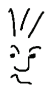
```
p{
  line-height:18px;
}

h2{
  line-height:18px;
}
h3{
  line-height:18px;
}
```

在规则中，可以用逗号分隔选择器，这样可以将一个规则应用到多个元素上：

```
p, h2, h3{
  line-height:18px;
}
```

更少的代码意味着更快的传输速度。但是要记住，这样的措施在让代码的体积变小的同时，也让组织代码变得困难一些。



小乔爱问……

可以用从网上找到的现成的 CSS 样式表吗

可以是可以，但是我不建议不加修改地直接使用它们。还记得 11.1 节的框注“不要只是复制代码!”吗？你可以看看 Eric Meyer 做的很受欢迎的预置样式表，[1]虽然这些看起来很好用，要记住这些预置的样式表是以普适性为目的设计的，其中可能包含了大量你用不到的元素。写样式时，你最好是做到自己给自己预置元素。

① http://meyerweb.com/eric/tools/css/reset/。

11.5.2　盒模型

HTML 中每一个块元素基本上都可以看成一个盒子，这个盒子的宽和高，外加内边距、边框粗细和外边距这些东西，决定了元素的尺寸。如果定义了一个宽 50 像素的盒模型（box model），为它的每一边都设定了 2 像素的内边距，还有 1 像素宽的边框，另外左右各有 5 像素的外边距，那么这个元素的宽将会是 50+2+2+1+1+5+5，也就是 66 像素。如果你打算将这样一个元素放进页面上一个 50 像素宽的地方，那么这种计算就显得非常有必要了。

不同类型的盒模型

浏览器标准的不统一又给开发者带来了问题。多少年来，IE 浏览器一直在用自己的独特算法来解析盒模型。它将边框和内边距当作内容的宽度计算，也就是说我们声明的区域里用来放内容的宽度只有 44 像素（50-2-2-1-1）。你可以想象，这会造成多大的问题。

IE6 和 IE7 采用了针对盒模型宽度的标准算法，但是只有在浏览器用标准模式呈现的时候才会激活这种算法。也就是说，它们默认的模式还是那个奇怪的模式，用到的是老旧的算法。你可以将这种奇怪的模式看成是向后兼容的模式。但是我还是会把它当成一种令人头疼的东西。

还好，用合适的文档类型（doctype）设置和字符编码设置可以让 IE 在标准模式下运行，在我们的 HTML 模板中已经解决了这个问题，所以不会被任何元素的宽度问题所困扰。

11.5.3　将内容居中

在 Photoshop 里制作布局的时候，我们将页面的宽度设为 900 像素。现在，我们了解了页面的宽度应该等同于浏览器的宽度，然而我们并没有设计一个随浏览器宽度缩放的流体布局，所以将利用 CSS 单独确定页面的宽。

index.html 中有一个 `id` 是 `page` 的 `div` 标签，它将页头、页脚和页面中部区域都囊括其中。我们会用下面的两个属性来确定这个 `div` 的宽度。

css_layout/layout.css

```
#page{
  display:block;
  width:900px;
  margin: 0px auto;
}
```

这一条规则将这个元素的宽设为 900 像素，还确定了它顶部和底部的 `margin` 是 0 像素，同时左右的 `margin` 会自动生成数值。

这个规则的冗长写法如下：

```
margin-top:0;
margin-right:auto;
margin-bottom:0;
margin-left: auto;
```

但是，在定义外边距的时候，我们可以用下面的简短模式：

```
margin:0px 5px 5px 0px
```

这行代码定义了在元素上方、右面、底部和左面的外边距。初见这个写法可能会觉得有些奇怪，然而你可以将之比作钟面的布局，12 在最上、3 在右面、6 在底部、9 在左面。

如果碰到我们这个例子里的情况，你甚至可以进一步压缩这行代码。你可以用 `margin:0` 这种形式来设定四个方位的外边距。你会常常看到这种缩写，因为这会减少字符的数量，从而节省页面的下载时间。

在把这些代码放到编辑器后，应该马上就能看到结果。现在页面应该是在浏览器里居中显示了。这也是一个保存文档的好时机。单击 CSS 编辑器中的保存按钮，将这个样式表保存在项目文件夹的 stylesheets 文件夹中，命名为 layout.css。

11.5.4　定义页头和页脚

页头和页脚的宽度都和页面相等，但是它们的高度和文字对齐方式不同。从样式页中可以看到页头的高度是 108 像素。将以下的代码加入 CSS 编辑器中。

css_layout/layout.css

```
#header{
  height:108px;
  width:100%;
}
```

这条声明定义了我们要求的高，同时将元素的宽设为 100%。你可能会认为这里的意思是占据整个屏幕宽，但实际上它的意思是占据它的父元素宽度的 100%，在这里就是 page 元素的 900 像素宽。

定义页脚的声明几乎是一样的，唯一的改变是需要将高度设置为 36 像素：

css_layout/layout.css

```
#footer{
        width:100%;
        height:36px;
}
```

11.6　将页面的单栏变成双栏

至此，我们的页面还没有什么特别的地方，但是好戏即将上演。CSS 的一大特色就是它

能够将元素从原来的页面“流”中剥离出来，并且将它们重新排位。我们的页面有一个侧边栏，以及和这个侧边栏并排的另一栏。我们会用一个叫做悬浮（floating）的简单技巧来实现这种效果。

11.6.1 文档流

在 9.4.6 节的框注“块元素和内联元素”中，你应该已经理解了元素的不同显示方式：块元素、内联元素或者不可见元素。了解这些差异，可以让你在布局中充分利用 CSS 的特点。例如，div 标签是一个块元素，浏览器在试图呈现这个元素的时候，会让它另起一行，并且占满这一行所剩下的所有空间。然而，可以利用 CSS 里的 display 属性来改变这个过程：

```
#page{
   display:inline;
}
```

display 属性有数个可以选择的值，我们现在只关心其中的三个：首先是 block，这个值可以让元素以块元素的形式被呈现；然后是 inline，这让元素以内联元素的形式呈现；另外还有 none，这个值将元素从文档中移除。

11.6.2 浮动

在读杂志、报纸或者教科书的时候，你会看到文字很精巧地浮动在图片周围。可以用 CSS 中的浮动属性来达到这个目的，也可以用这个方法来让元素并排排列，让它们看起来像两列文字的布局。

将一个元素设定为浮动，意味着将这个元素从文档的“流”中单独拿出来，然后页面中剩下的内容会将这个元素围绕起来。如果让两个挨着的元素都变成浮动，你就得到了所需要的双栏布局的效果，当然还需要给两个元素分别设定宽度。

看看这个简单的结果，有两个拥有内容的 div 标签，其中一个是被提出来的“盒”，另外一个是剩下主要文字内容：

css_layout/float_wrap.html

```
<div class="callout">
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>
</div>
<div class="content">
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat. Duis aute
```

```
    irure dolor in reprehenderit in voluptate velit esse cillum
    dolore eu fugiat nulla pariatur. Excepteur sint occaecat
    cupidatat non proident, sunt in culpa qui officia deserunt
    mollit anim id est laborum.</p>
</div>
```

可以让主要文字内容环绕在被单独提出的文字四周，只需要为那个单独的“盒”加上浮动属性：

css_layout/float_wrap.html

```
.callout{float:left; width:108px;}
```

得到的效果看起来就跟图 11-6 一样。然而，如果给相邻的两个区域都加上 `float` 属性，它们就会以列的形式排列起来，如图 11-7 所示。

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

图 11-6 一个单独的浮动元素让其他内容围绕在它四周

css_layout/float_columns.html

```
.callout{float:left; width:108px;}
.content{float:left; width:400px;}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

图 11-7 两个相邻的浮动元素形成了两列

为了创建侧边栏和侧边栏区域，需要将侧边栏和主要区域都设为浮动。在 HTML 代码中，这两个区域被一个叫做 `middle` 的区域所包含。

将 `middle` 区域设置为 100%宽。如果你不这样做，区域中的元素可能不会如你所愿的那样扩展：

css_layout/layout.css

```
#middle{
        width:100%;
        float:left;
}
```

然后定义侧边栏。根据样式页，侧边栏的宽是 306 像素（注意网格线!）。`float:left` 属性让这个元素处于最左边的位置，并且其他元素会环绕在它周围。在 CSS 编辑器中写下这些代码后，你会立刻看到侧边栏浮动到了左边：

css_layout/layout.css

```
#middle #sidebar{
  width:306px;
  float:left;

}
```

当然，你并不希望 `main` 区域环绕在侧边栏周围，你需要这两个区域看起来是两栏。最简单的解决方式就是让 `main` 区域也是向左浮动的，然后给这个区域设置一个合适的宽。这个宽度应该是页面宽减去这一行中所有其他元素的宽，以及它们的外边距、边框或者内边距。别傻算了，答案应该是 594 像素。跟那张从 Photoshop 中导出的意大利面的照片是一样的宽，这个图片充满了整个 `main` 区域：

css_layout/layout.css

```
#middle #main{
  width:594px;
  float:left;

}
```

现在我们看到，所有在 Photoshop 中花费的时间都带来了回报！

这两条 CSS 规则都有缩小范围选择符。当选择符中出现空格的时候，我们实际上是在缩小选择符的范围。在`#middle #sidebar` 这个例子里，选择符的意思是说：`id` 是 `sidebar` 的元素，是 `id` 为 `middle` 元素的下级。`id` 在页面中是唯一的，所以缩小范围在这里并没有什么显著的效果，只是为了组织代码和可读性准备的。但是在后面的例子中，缩小范围是个很有用的技巧，如下所示：

```
a{color:#339;}
#sidebar a{color: #fff;}
```

有了这些规则，在侧边栏中的链接的颜色会有别于页面中的其他链接。如果侧边栏的背景颜色比主区域要暗很多，你就应该试试这个方法。

11.6.3　背景颜色和浮动

如果一个 div 中的所有子元素都离开了文档流，Firefox 和其他遵守标准的浏览器也不会为这个 div 添加任何背景颜色和边框。实际上，这个 div 的高度会“坍塌”——也就是说这个 div 的背景图片、边框或者背景颜色都是不可见的，看看如下代码：

```
<div id="container">
  <div id="col1">
    <p>foo</p>
  </div>
  <div id="col2">
    <p>bar</p>
  </div>
</div>

#container{
  background-color:#ffe;
}
#col1{
  float:left;
  width:400px;
}
#col1{
  float:left;
  width:400px;
  background-color:#eee;
}
```

在这个例子中，你看到的是两竖栏被包含在一个容器中，每个竖栏都是浮动的，这就会造成容器坍塌，也就是说容器中的背景色不会被显示。解决方法虽然简单，但是也没有那么明显：你需要将容器也变成浮动的。一旦容器变成浮动的，背景色也就可以显示出来了。[①]

另外一个解决方案是在容器 div 关闭之前，加上一些其他的元素，比如一个换行符，然后为这个元素加上清除浮动的属性：

```
<div id="container">
  ..
  <br class="clear" />
</div>

...
.clear{
  clear:both
}
```

① 不要太担心 IE 中不太引人注意的“双外边距”的 bug。这个 bug 是说两个相邻的浮动元素会产生多余的外边距。第 15 章会讲到这个问题。

这两种方法效果都很好，但是后面的那个方法需要你写额外的标签来解决问题。如果你在自己的页面设计中遇到了这个问题，采取什么方法都可以。

缩进选择符

如果你将相关的选择器分成组并且缩进元素，那么 CSS 代码的可读性会好很多，例如以下的代码就很容易阅读：

```
#navbar {
  height: 36px;
  margin-bottom: 24px;
}
  #navbar ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
    #navbar ul li {
      float: left;
      margin-right: 20px;
    }

#middle{
  width:100%;
}
```

但是接下来的这些代码就没有上面的容易读懂了：

```
#navbar {
  height: 36px;
  margin-bottom: 24px;
}
#navbar ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
#navbar ul li {
  float: left;
  margin-right: 20px;
}
#middle{
  width:100%;
}
```

虽然区别看起来微不足道，但是这种缩进带来的视觉上的引导可以让你更容易查找想要寻找的东西。

11.7　为内容加上外边距

基本的结构已经差不多了，但是看起来页面还是不太可读。我们去掉了大多数元素的外边距，现在该将外边距加回去了。从侧边栏中的元素开始，把所有元素的外边距都设为 18 像素的倍数。

快速格式化侧边栏元素

所有侧边栏中的元素都有相同的结构：都是身处父元素中的子元素。都有标题，紧接着标题就是内容。在 11.5 节中，设定了标题的默认外边距（18 像素）。现在需要单独为这些区域定义外边距：

css_layout/layout.css

```
#browse_recipes, #popular_ingredients, #search{
  margin-left: 18px;
  margin-right:18px;
  margin-top: 18px;

}
```

这几行代码应该就可以解决侧边栏的外边距了。我们将侧边栏中的元素都加上左边、右边和上方的 18 像素外边距，用来规整这些元素。这里最需要注意的是三个元素共用了这些声明。在任何可能的时候，只要看起来合理，你都应该这样写 CSS。[①]

11.8　主区域

主区域由一张意大利面的图片、一个双栏布局和另外一个单栏布局组成。其中意大利面的图片不需要任何 CSS 样式，至于其他的元素，你可以使用刚刚学到的模式来完成。唯一需要特别注意的是除了主区域里的那张图片，其他所有的内容都有一个 18 像素的左外边距。但是我不会给主区域设置 18 像素的外边距，而是将外边距设置给 Get Cookin’和 Latest Recipes 两个区域。

11.8.1　主区域文字

主区域文字需要浮动至登录按钮的左侧，我想通过侧边栏的学习，你应该已经知道怎么样去做了。但是这里要注意中间区域的宽。Get Cookin’区域和两个按钮所在的区域是等宽的，而主区域的宽度又是 594 像素。起初你可能认为只要将 594 像素均分给这两个区域就好了。但实际上这

① 当然这会增加你的工作量，所以你要在机智和可读性之间做出权衡。

种做法是错的，因为你忽略了 18 像素的左外边距。记住，外边距是要算在实际宽度中的（至少在大部分普通浏览器中是这样!）。所以正确的算法应该是：(594−18)/2=288。

因此，Get Cookin’区域的代码是这样写的：

css_layout/layout.css

```
#main_text{
  float:left;
  width:288px;
  margin-left:18px;

}
```

11.8.2 注册按钮区域

注册按钮区域跟主区域非常接近，你可能想让它们都共用一套样式，但实际上你还是需要一些细微的修改。首先，在样式页中，注册按钮出现在意大利面图片下方 36 像素的位置，你需要给它加上 36 像素的顶部外边距。另外，按钮在这一栏是居中显示的，因此你可以用 `textalign:center` 来实现这个效果，它可以让这个区域内的所有元素都居中，包括段落甚至是 `div`：

css_layout/layout.css

```
#signup_login{
  margin-top:36px;
  float:left;
  width:288px;
  text-align:center;
}
```

按钮

按钮也需要一点点样式。在默认情况下锚点和图片都是内联式的元素，这意味着它们是并排显示的。而在当前的例子中，我们希望按钮是显示在其他元素之上，并且有 18 像素的外边距将它们隔开。因此，我们只需要将锚点标签的 `display` 方式从 `inline` 改为 `block` 就可以了，另外，需要为它们加上 18 像素的底部外边距：

css_layout/layout.css

```
#signup_login a{
  width:100%;
  display:block;
  margin-bottom:18px;
  float:left;
}
```

在这种情况下，你肯定希望这些样式只局限于某个范围之内的。如果没有这样做，那么整个页面的链接都会受到它的影响，而你显然不希望发生这种事情。

11.8.3 最新菜谱

至此，完成主区域中最后一块地方的格式化应该是一件很容易的事情了，因为你所需要做的只是重复使用已经学会的技巧。每个菜谱的标题都会有18像素的缩进，然后每个菜谱的描述会在此基础上再缩进18像素。每个菜谱的标题都是用 `h3`标签标记的，而它们的描述则是一些简单的段落。

清除浮动

只要一个元素被设置为浮动的，那么它之后的所有元素都会环绕它显示，除非你让某一个元素回到正常的文档流。这就是所谓的*清除浮动*。当你有一个单栏布局紧跟这一个双栏布局的时候，这个技巧就派上用场了。在某个区域的 CSS 规则中用上 `clear:both`，可以让这个区域回到普通的文档流中。

用缩小范围的定义方式，只需要一点点代码就可以弄好了：

css_layout/layout.css

```
#latest_recipes{
   clear:both;
   margin-left:18px;
   margin-right:18px;
   margin-top:18px;
}

  #latest_recipes h3{
    margin-left:18px;
  }

  #latest_recipes p{
    margin-left:36px;
  }
```

这些代码不但设置了缩进，而且也让这个区域回到了文档流中。

11.9 回到页脚

虽然现在页脚里的东西看起来都是正常的，但是为了以后的进一步改进，你肯定需要为它加上一些 CSS 代码。页脚区域是紧跟在向左浮动的中间区域之后的。在 11.8 节，你学会了如何清

除浮动效果，让元素回到原来的文档流中。在上面那个例子里，我们利用了“最新菜谱”区域来达到这个目的，但是在其他页面中你可能就不会有这样的机会，因此我总是会建议在页脚做清除浮动的工作。

11.10 小结

本章的内容比较多。至此，你应该学会了如何利用 CSS 实现一个简单的双栏布局效果。首页已经布置的差不多了，现在是为它添加一些色彩的时候了。

第 12 章 利用覆盖法替换各区域中的标题

12.1 什么是覆盖法

覆盖法并不是用图片替换文字。顾名思义，这个方法会将一个新建的图层放在文字的上方，并且在新图层中放置一张图片。在讲解 CSS 的时候，我们还没有提到层的概念，但是基本上 CSS 允许你把元素放置在任何地方，前提是你要了解这种摆放对页面其他部分的影响。

另外，还有一些简单的替换法，像 Fahrner 图片替换法（Fahrner Image Replacement）所采取的办法是将文字设置为 `display:none`，让文字不能显示，然后再把 CSS 设置的图片放在包含文字的标签里面。然而，新版本的屏幕阅读器开始检查 CSS 属性了，这将导致文字被隐藏起来，不能被用户听到。

覆盖法则避免了这个问题。屏幕阅读器不会载入图片，文字也可以被朗读出来。另外，当样式表被关掉之后，页面的显示看起来也还不错。

12.2 为覆盖做准备，调整 HTML

为了让覆盖法正常工作，在需要“覆盖”的地方加上一对 span 标签。打开 index.html 文件，找到 Search Recipes 这个标题。然后，将里面的内容略做修改，在 h2 标签闭合前，加上一对 span 标签：

```
<h2 id="search_header">Search Recipes<span></span></h2>
```

然后用 CSS 在 span 标签里载入图片，再将 span 从正常的文档流中抽离出来，让它位于文字的上层。保存文件，并且再次进行验证，保证标记都是合法的。

12.3 覆盖文字

首先要做的事情是将 h2 标签变成一个有长度和宽度的容器，以便能够装下图片：

```
#search_header{
   margin:0; padding:0;
   position:relative;
   width:180px; height:36px;
   overflow:hidden;
   }
```

然后把内联元素 span 变成块元素，这样就能为它设定宽和高了。接下来给它加上 position: absolute 这个属性，这样这个元素就能用相对于 h2 元素的坐标来确定位置了。我们刚刚也给 h2 标签加上了 position:relative 属性。

把 span 的 margin 和 padding 值都设为 0，以防万一。最后把图片以背景图的形式载入到 span 之中：

```
#search_header span {
   display:block;
   position:absolute; left:0; top:0; z-index:1;
   width:180px; height:36px;
   margin:0; padding:0;
   background:url("../images/search_header.gif") top left no-repeat;
}
```

12.4 替换所有其他标题

然后将这个步骤重复几次，用同样的方式处理侧边栏中的其他标题。记得要在 HTML 里加上 span 标签。为了减少样式表中的代码量，你可以用选择符分组法，将样式相同的规则组合在一起：

coverup/style.css

```
#search_header,
#browse_recipes_header,
#popular_ingredients_header{
   margin:0; padding:0;
   position:relative;
   width:180px; height:36px;
   overflow:hidden;
}

#search_header span,
#browse_recipes_header span,
#popular_ingredients_header span  {
  display:block;
  position:absolute; left:0; top:0; z-index:1;
  width:180px; height:36px;
  margin:0; padding:0;
}
```

```
#search_header span{
  background:url("../images/search_recipes.gif") top left no-repeat;
}

#browse_recipes_header span{
  background:url("../images/browse_recipes.gif") top left no-repeat;
}

#popular_ingredients_header span{
  background:url("../images/popular_ingredients.gif") top left no-repeat;
}
```

在完成侧边栏里的替换之后，对主区域中的两个标题也做同样的处理。需要做细微调整的地方是将高和宽改成 198 像素和 54 像素，我相信你自己应该可以完成这次替换：

coverup/style.css

```
#get_cooking, #latest_recipes_header{
  margin:0; padding:0;
  position:relative;
  width:198px; height:54px;
  overflow:hidden;
}

#get_cooking span, #latest_recipes_header span  {
  display:block;
  position:absolute; left:0; top:0; z-index:1;
  width:198px; height:54px;
  margin:0; padding:0;
}

#get_cooking span{
  background:url("../images/get_cookin.gif") top left no-repeat;
}

#latest_recipes_header span{
  background:url("../images/latest_recipes.gif") top left no-repeat;
}
```

12.5　替换链接

你可以用相似的方法处理超链接，但是应该用 span 将超链接中的文字包裹起来，而不是用空的 span。现在用页头中的 Foodbox 图片做个例子，标记应该这样修改：

coverup/replacedheader.html

```
<h1><a id="foodbox_header" href="/">Foodbox<span></a></h1>
```

CSS 代码则跟上一个例子差不多，但是要记得 display:block 属性需要同时应用在 span 标签和 a 标签之上，这样才能创建可点击的图片。因为超链接和 span 元素默认都是内联元素，而

没有办法给它们设定宽和高。

相应的样式代码应该是这样的：

coverup/stylesheets/replacedheader.css

```
#foodbox_header{
   margin:0;
   padding:0;
   position:relative;
   width:486px;
   height:90px;
   overflow:hidden;
   display:block;
}

#foodbox_header span {
  position:absolute; left:0; top:0; z-index:1;
  width:486px;
  height:90px;
  margin:0; padding:0;
  background:url("../images/banner.png") top left no-repeat;
}
```

透明

如果你在我们的页面上做了以上的尝试，就会发现覆盖法其实根本覆盖不了什么东西。文字会通过图片的透明部分显现出来。为了修正这一错误，需要借用另外一种替换法：Langride 或 Leahy 图片替换法（LIR）。[1]你需要将 `span` 的高度设置成 0，再将它的顶部 `padding` 设置成跟图片一样的高度。因为一个盒模型的高度是它自己的高度外加上下部分的 `padding`，所以这样一改，我们就得到了想要的结果，而文字则被隐藏起来：

coverup/stylesheets/replacedheadertransparency.css

```
#foodbox_header{
   margin:0;
   padding:0;
   position:relative;
   overflow:hidden;
   display:block;
   width:486px;
   /*   height:90px; */
   height:0;
   padding-top:90px;
   font-size:10px;
}
```

① http://www.kryogenix.org/code/browser/lir/。

这段代码也减小了字体的大小，避免文字的上半部分超出范围。

12.6　这种方法的缺陷

你最可能注意到的缺陷可能是，尽管得到了漂亮一些的字体和更符合网格的对齐，但是在 HTML 文档中添加了额外的标记来让这个方法生效，同时还引入了不少额外的 CSS 标签。至于是否要在项目中用这种方法，还是取决于你的决定。

另外，屏幕阅读器曾经让其他的图片替换方法失效，所以你需要时常注意这一点。换句话说，这个方法可能不会一直有效。另外一个替代方法是嵌入图片，并且为之加上 `alt` 属性。但是这样做就可能让搜索引擎忽略你的标题。在读完第 18 章后，你可以了解到更多内容。

12.7　小结

本章介绍的方法让你学会了一个“搜索引擎友好”的方法来保留标题区的字体。另外，这个技巧还能用在其他元素身上。

第 13 章

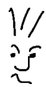
添 加 样 式

你已经掌握了如何用 CSS 来确定元素的位置和页面布局，并且已经在页面上添加了一些图片。接下来，需要看看怎么让页面变得更漂亮一些。读完本章后，你具备的 CSS 基础就应该可以让你独自完成自己的下一个项目了。

另外，本章有些地方你应该是比较熟悉的了。以前你可能用过 CSS 设置字体，或者用它做过一些基本的视觉上的调整。本章将会从字体和颜色开始学习，同时还会结合一些你在第 11 章里学到的东西。

13.1　设置字体和颜色

作为程序员，对 CSS 最恼火的事情可能就是它没有变量这个概念，CSS 不会提供可以定义并且能重复使用的变量。如果在页头和页脚中都要用到 16 进制代码#FFE500，那么就不得不将这个代码放在两条不同的 CSS 规则当中，或者是尽自己最大的努力用逗号分组分隔符的方式来共享规则。我觉得一个比较好的方法是把关于颜色的声明都放在样式表中靠前的位置，这样找起来就不会很困难。



小乔爱问……

可以动态地生成样式表吗

当然可以，就像你也可以动态生成 HTML 一样。你需要做的是让你的网站后台对关于样式表的请求做出正确的反应。这是一个很高级的方法，同时你还需要设定一些缓存机制，避免程序为每个请求都生成一套样式表。

这个想法听起来很有诱惑力，但是你需要考虑 CSS 不存在变量这个特点的影响有多大。更换颜色的频率高不高？重复颜色的频率呢？太多数情况下，在服务器端生成这些东西的代价会很高。就跟其他类似问题一样，这取决于你的具体情况。

13.1.1 风格手册的重要性

很多公司都有一本风格手册（style guide），一般是公司里做品牌定位的部门发布的。这本手册一般包含的内容是关于公司出版物中的颜色和字体的要求。如果存在这样一本风格手册，那么在做东西的时候应该尽可能遵守其中的规定。Foodbox 并没有这样的一本手册，大多数小公司也没有。在本书中，你差不多会完成所有的关于风格手册的基础工作，这将对你未来的项目十分有帮助。

风格手册通常详细规定了排版布局、字体和颜色的样式，同时也会有内容写作风格的相关条款。比如，如果我很讨厌在网页中用到“单击”这个总是让我如鲠在喉的词，那么我会在风格手册里面加上一条类似的条款，用来防止任何人在某个文案中用到这个词。在风格手册中，你还可以规定应该使用单词“and”而不是“&”符号，或者是否需要为链接加上下划线，等等。

建立风格手册的目的是为了保持公司的所有出版物和流通印刷品在视觉呈现上保持一致。

在本章中写的样式表，其实就是一种风格手册的具体代码实现。

目前已经用图片实现了不少字体，但是在 CSS 中并没有多少关于字体的声明。

一些关于 class 名字的说法

“用具体的规则作 class 的名字”，这个想法听起来可能很诱人，但是你最好不要这样做。一直以来我见过很多这种 HTML 代码：

```
<p class="red">An error has occurred</p>
```

初见这段代码的时候，你可能觉得它很漂亮，而且可以清楚地知道设计师希望这段错误提示是红色的。

但是实际上这段文字最终可能是绿色的，因为可能会有另外一个人插手其中，觉得错误提示不应该是红色的，因为红色不太好看，会吓到用户。所以更好的 class 名应该是这样的：

```
<p class="critical_warning">An error has occurred</p>
```

然后你可以为这个 class 加上符合风格手册的样式。class 名中应该包含的是关于“目的”的信息。

在 style.css 文件中加入以下代码：

css_style/stylesheets/style.css

```
body{
  font-family:Arial, Helvetica, sans-serif;
  font-size:12px;
}
```

另一方面，文字颜色的设置可能需要耗费更多精力。我们不仅要定义各个区域中的文字颜色，还要确定每个链接的颜色。

13.1.2 伪类

在样式表中加入下面的代码：

css_style/stylesheets/style.css

```
#header, #footer{background-color:#FFE500}
#middle{background-color: #ffdd7f}
#main{background-color:#fff8e4}

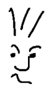
a{color:#4d3900; text-decoration:none;}
a:visited{color:#806f40;}
a:hover{color:#807940; text-decoration:underline;}
```

上面关于颜色的 CSS 声明中，有两条是 `a:hover` 和 `a:visited`。这种声明称为伪类(pseudo-classe)。到目前为止，你所见到的样式应用都是基于元素在文档树中的位置的，但是 CSS 标准也允许基于非文档树信息的样式应用，比如说基于浏览器提供的信息的样式。浏览器默认会把访问过的链接和未访问过的链接用不同的颜色区分。许多年以来，这个特性已经变成了可用性的基础，用来帮助用户辨认已经看过的信息。本来，已访问链接的颜色是应该在每一页的 `body` 标签中定义的，但是如果用 CSS 伪类来定义，只需要在样式表中标明就可以了。

`:hover` 伪类会捕捉鼠标事件，这是个十分强大的特性。你可以用它来改变鼠标停留在链接上时链接文字的颜色。我们的例子就是这样做的。同时，你还可以用它来制作下拉菜单，或者用它来展示页面的另外一个区域。

然而，浏览器也会影响这些功能。在 IE6 中，`:hover` 只对链接有效，而我们可以在其他浏览器中除链接之外的其他元素上使用这个功能，从而创造一些引人注目的效果，比如说图片翻滚或者表单域突出显示等。

上面的示例还去掉了链接的下划线，设定只有当鼠标悬停在链接之上的时候才会出现下划线。注意我们是如何移除下划线，又是怎样在伪类中加上下划线的。



小乔爱问……

为什么要移除下划线，这难道不是一种损坏可用性的做法吗

这是一种可能，用户会依据下划线来判断文字是否可以单击。然而，时代在进步，用户

们的习惯也与时俱进，他们开始学会寻找其他的线索，如不同颜色的文字。但是，去掉下划线是一种冒险，如果标题的颜色和正文文字不同，这些单独文字的颜色可能会让用户困惑，或许你该为标题考虑一种差别不是很大的颜色——我想你已经知道我在说什么了，用法应该取决你的目标用户，取决于他们是怎么想的。我的想法是现在你已经知道具体的技术，那么什么时候该用什么东西，应该由你来决定。

13.2　标签云

以这个构架完整的文档为基础，可以用少量的代码来实现标签云的效果。在 HTML 文件中，用 span 标签包裹起了每个标签，并且给它们赋上了从 level_1 到 level_5 的不同 class。越欢迎的标签，level 数越小，反之亦然。为了实现这个效果，需要为重要的标签加上大而明显的样式，为不那么受欢迎的标签加上小而不太明显的样式：

css_style/stylesheets/style.css

```
.level_1, .level_2, .level_3, .level_4, .level_5{
        margin-bottom:18px;
        margin-left:18px;
        line-height:36px;}
.level_1{font-weight:bolder; font-size:20px;}
.level_2{font-weight:bold; font-size:18px;}
.level_3{font-size:16px;}
.level_4{font-size:14px;}
.level_5{font-size:12px;}
```

13.3　搜索表单

页面上就剩搜索表单没有弄好了。跟搜索框旁边的那个漂亮图片按钮比起来，表单显得有点怪。我们需要给搜索框设定高度、宽度和边框，而且还需要设定里面的字体。我们需要将这些关于高和宽的东西定义在样式表中，而不是单独地分离出来。style.css 里所有的东西应该形成一个整体：

css_style/stylesheets/style.css

```
#search_form #search_keywords{
        width:200px;
        float:left;
        border:1px solid #000;
        height:16px;
        padding:0;}
```

要特别注意这里的像素计算。行高是 18 像素，但是搜索框的高只有 16 像素，因为它的上下分别有 1 像素的边框。

13.4 页脚

在页脚中，文字都是居中的，而且需要在样式表中定义的 36 像素高的区域内安排两行文字。居中文字很简单，你只需要在页脚元素中应用 `text-align:center` 这个属性就好了。

关于安排那两行文字的方法也有不少。大多数开发者的第一反应是在 HTML 中把段落标签 `p` 删除，在两行文字间用 `br` 换行标签，但是这种方法让页面在语义化上有了一点缺陷。调整段落标签的样式会更加简单一些。我们可以将页脚里的段落 `margin` 设为 `0`，这样段落之间就没有间隙了。而且，这种方法还为我们留出了很大的灵活性。

在样式表中加上：

css_style/stylesheets/style.css

```
#footer{text-align:center;}
#footer p{margin:0;}
```

13.5 清理零散的角落

首页上的工作基本上完成了，但是还需要解决一些没有对齐或是不太精确的地方。首先，页面上的注册和登录按钮四周有边框，需要去除这些边框。其次，页面上的黄色没有像预想的那样随着屏幕大小拉伸。我们会很快把这些东西修复好！

13.5.1 去掉图片的边框

带有链接的图片会被自动加上一圈边框，用来提示用户这个图片是可以单击的。在 HTML 的远古时代，开发者可以在 HTML 标签中加入 `border="0"`属性来达到这个目的，但是这种方法将设计和内容混杂在一起了，我们可不希望做这种事情。同时 doctype 也认为这种方法是非法的。还好我们还有别的简单方法：在样式表里关掉边框。

在样式表中加入以下代码，这些代码可以去掉所有图片的边框：

css_style/stylesheets/style.css

```
img{border:none;}
```

刷新页面后，绿色的标签就会消失。

13.5.2 拉伸Banner里的颜色

有很多技巧可以达到这个目的。比如可以将页头从名叫 page 的 div 容器中拿出来，将它放在另外一个 div 中，让这个 div 的宽度值是 100%，同时保持页头的宽度跟 page 容器一样。代码如下：

```
                <body>
                        <div id="header_wrap">
                                <div id="header">

                                </div>
                        </div>
                        <div id="page">
                                ...
                        </div>
        </code>

    <p>Some CSS associated with this code would look like this:</p>

        <code language="css">
               #headerwrap{width:100%;
                 float:left;
                 height:108px;
                 background-color:#FFE500;
               }

               #header, #page{
                 margin:0 auto;
                 width:900px;
              }
```

这个方法很常见，并且对那种只有简单的单一颜色很有效。但是如果有很多页面要改，而又不愿意去调整每个页面，又该怎么办呢？或者如果觉得新加一个容器会让代码看着不够聪明，这时要怎么做呢？可以用重复背景图片的技巧来达到相同的目的。

在 Photoshop 中打开样式文档，选中切片工具。

第一步要做的是从页面中切下一小片图片，包括了整个黄色 Banner 的高度以及一点点的白色。在页面的左上角创建一个宽 1 像素、高 128 像素的切片。你既可以手工将切片缩小到这么小，也可以在设置切片名字的时候调整它的宽度，并且将它命名为 background。设置好之后，将这个切片以 GIF 格式导出到 images 文件夹。

然后在样式表中为 body 标签加入如下代码，让这个小切片水平重复排列：

css_style/stylesheets/style.css

```
body{
```

```
        background: #fff url('../images/background.gif') repeat-x;
}
```

当你在 Firefox 中预览页面时，它看起来应该和图 13-1 一样。



图 13-1 在 Firefox 中的 Foodbox 首页

13.6 小结

经过很长时间的锻炼之后，你终于做到了这一步，亲手写出符合规则的代码，适用于 Firefox 浏览器（还有 Safari，去看看吧）。但是，作品还没有完工，除非你把它放在其他环境中测试一下，特别是那个用户甚广的由微软生产的浏览器。

第 14 章

制作打印机友好的页面

Foodbox 是一个关于菜谱的网站，我们期待用户会打印出他们在网站上找到的菜谱。这本可以用后台程序实现支持打印机的功能，但是在本章中，我将向你展示如何只用 CSS 来改变页面的外观，让它满足用户的打印需求。

14.1 准备工作

当用户打印一个网页的时候，他通常只关心页面上的有效信息。[①]在打印页面的时候，像侧边栏、导航栏、一些图片、背景颜色甚至花哨的标题等元素都没什么用，将它们打印出来反倒是浪费墨水和纸张。所以，当你需要做一个打印友好版本的页面时，首先要考虑的是如何去掉这些元素。

当在页面上链接 layout.css 和 style.css 文件的时候，为它们指定的是只在屏幕显示的情况下调用。这就意味着在打印的时候，页面是没有样式的，也就是说可以从头开始专门为打印功能设计一套样式表。这样也就不用担心因为出现样式冲突或者覆盖而让用户浏览器的默认样式趁虚而入的问题了。

14.2 链接打印用样式表

新建一个 print.css 文件，将它保存在 stylesheets 文件夹中。打开 index.html 文件，在 `style` 标签之后加入以下代码以链接这个文件：

css_print/index.html

```
<link rel="stylesheet" href="stylesheets/print.css"
      type="text/css" media="print" charset="utf-8">
```

这次将 `media` 类型设置成 `print` 而不是 `screen`。在用户进行打印预览操作或者将页面发送

① 另一方面，客户可能需要打印出整个页面。这样他们才可以在页面上写写画画，然后给你提意见。但是客户并不是网站的主要用户。

到打印机的时候，现代浏览器会启用这个样式表。在制作的工程中，你可以利用浏览器的打印预览功能来测试样式的好坏，同时你应该尽可能在真实的打印机上做一些测试。

14.3 去掉不需要的元素

下面研究一下主页，看看哪些东西是不需要打印出来的。首先，侧边栏和标签云是不需要打印出来的，所以可以很放心地将它去掉。那张意大利面的图片也是可有可无的，另外还有登录和注册按钮。

不用担心那些主页上的颜色，因为在打印的时候不会调用相应的样式表。

那些服务条款和隐私政策的链接也是无用的，一并去掉。

在写 HTML 文档的时候，可以给上述的区域都加上了唯一 id 以便引用。现在，需要做的只是将这些区域的 display 属性设置成 none 就好了，在 stylesheets/print.css 中加入如下代码：

css_print/stylesheets/print.css

```
#sidebar, #main_image, #signup_login,
#privacy_and_terms, .noprint{
  display:none;
}
```

Noprint class

在上面的选择符中，我加入了一个叫做 noprint 的 class，你可以把这个 class 用在页面上所有打印时需要隐藏的元素身上。例如，你可以在 Logo 元素的 HTML 代码中加入 class="noprint" 来隐藏这个元素。这种做法在完美主义者眼中可能是模糊内容和显示这两者界限的举动，但是在使用后台脚本生成动态元素的时候，这是一种高效率的做法。

将一个元素设置成 display:none 后，它不仅仅是“不可见”的状态，这个元素同时还被移出整个文档。

14.4 设置外边距、宽度和字体

在为屏幕显示设计布局的时候，曾经关掉了所有浏览器默认的样式，然后定义了自己的外边距、行高和字体大小。在这里我们会使用一套同样的程序，只是会用点（point）来代替像素，因为打印机使用的单位是点。

在样式表中加入以下内容：

css_print/stylesheets/print.css

```
body, p, h1,h2,h3,h4,h5{margin:0; padding:0;}
p, h1,h2,h3,h4{line-height:18pt;}
p{font-size:12pt; margin-bottom:18pt;}
h1{font-size:18pt;}
h2{font-size:16pt;}
h3{font-size:14pt;}
```

14.4.1 页面外边距

你可能会注意到，我们已经将页面的 margin 设置为 0，因为在打印的时候出现的外边距大多依赖于操作系统的打印机驱动程序，而在 CSS 中增加更多外边距所导致的结果，是文档内容距离页边的外边距空间比你想象中的要大得多。

14.4.2 选择一个字体

很多浏览器都会默认使用衬线字体，这种字体在打印文档中更容易阅读。关于这个字体，我在 4.2 节中已经讨论过了。

在打印用样式表中加入如下代码：

css_print/stylesheets/print.css

```
body{
  font-family: Baskerville, Times New Roman, Times, serif
 }
```

上面那段代码确定了在 body 中使用 Baskerville 字体，另外还设置了两个备选方案。这条规则将被应用在 body 中的每一个元素上，除非它们有其他的自定义字体。

在打印的时候处理图片

到目前为止，图片一直都是用像素来做单位的，在打印的时候，如果图片不能被隐藏，那么有两种方法可以解决这个问题。第一种方式是用 CSS 将图片的高和宽改成合适的比例，第二种方法是改变图片周围的文字位置，不让那些文字环绕图片显示。

在打印用的样式表中改变图片大小可能会有些风险。如果增大图片则让打印结果中图片的像素颗粒更明显。但是如果图片具有可以用来引用的 ID，那这个方法还是可行的。

第二个选择则要好一些，用 CSS 来重新组织页面，让图片不再浮动。

记住，页面上的图片可能根本就不适合打印，因为它们只有 72dpi。如果希望在网站内容中加上高分辨率的图片，你可能需要在服务器端做一个 PDF 生成的功能，并且使用高质量的可以被打印的图片。

14.4.3 加上一个分隔符

现在没有了让我们分辨不同区域的颜色，所以只能用一条细的黑色线段为页头加上一个下边框，区分开页头和内容。

css_print/stylesheets/print.css

```
#header{border-bottom:1px solid #000;}
```

你可以用类似的规则来制作水平分隔线，用来分开每个区域，而没有必要将这些分割线当成页面内容放进文档。

14.5 搞定链接

人们读打印版页面的时候没有办法知道那些链接将会链到何方，这里将利用一段比较“先进”的 CSS 代码让上述情况得到改进。

在样式表中加入如下代码：

css_print/stylesheets/print.css

```
#main a:link:after, #main a:visited:after {
  content: " (" attr(href) ") ";
  font-size: 90%;
}
```

在此，使用了 CSS 抓取 href 属性，并把它放在了内容当中。这个小技巧在任何地方都是可行的，除了 IE8 以前的 IE 浏览器，这些浏览器会忽略这个规则。

这样，样式表就完成了，在打印页面的时候，这个页面看起来会跟图 14-1 一样。

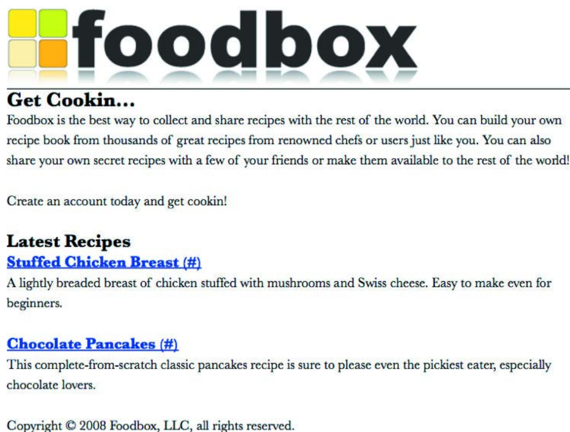


图 14-1 在打印样式表下呈现出来的首页

强制分页

用样式表你可以在打印的时候强制分页。假设要打印出一个带标记的菜谱文档，如下所示：

```
<div class="recipe">
   <h2>Bacon Explosion</h2>
   <ul>
     <li>2 pounds thick cut bacon</li>
     <li>2 pounds Italian sausage</li>
     <li>1 jar of your favorite barbeque sauce</li>
     <li>1 jar of your favorite barbeque rub</li>
    </ul>
    <p>.....</p>
</div>
<div class="recipe">
  <h2>Amazin' Bacon Burger</h2>
  ....
</div>
```

如果想要每个菜谱在打印的时候都单独占一页，那么可以加入如下代码：

```
.recipe {page-break-after: always;}
```

然后，如果对这个 Bacon Explosion 的菜谱感觉不错，你可以去亲自试试看！①

14.6　还要应付不习惯专有打印样式的用户

有些用户可能希望打印版的页面就是原有的网页。实际上我遇到过这种客户，他们看到打印版页面跟原来的页面不一样就会很沮丧。

为了稍微安抚一下这种情绪，你可以在页面上的某个位置加上一个 Print Contents Only（只打印内容）链接。这个链接看起来可能是这样的：

```
<a href="#" onclick=" window.print(); return false">Print Contents Only</a>
```

这个链接用到了 JavaScript，把动作和内容混杂在了一起，你或许也应该考虑用一段不起眼的代码来添加这段链接。

你也可以将 layout.css 和 style.css 用作打印样式，在打印时加载它们。但是有的浏览器在对付长而浮动的元素时会有问题，所以可能需要用打印样式来覆盖一些样式规则。但是我并不推荐这样做，因为我个人认为，比起卷入丑陋的“覆盖样式”中，教会客户和用户接受这种新的打印

① http://www.bbqaddicts.com/bacon-explosion.html。

形式要简单得多。一旦你的用户知道了必须用一套新的打印用页面布局的原因，像节省墨水和纸张、让注意力集中在内容上等，他们甚至可能会欣赏这个主意。

14.7 小结

打印用样式表的实现过程是很容易的，并且这个做法能极大地增加用户体验，只要保证过程和结果都是简洁的。你能用这些样式把页面内容变得整齐、易读。你可以用相同的技巧来让页面适应手机、投影仪（需要浏览器支持），甚至是屏幕阅读器一类的辅助性软件。当然，这一切都是建立在对页面内容正确标记的基础上的，并且还要注意内容和显示的分离。

Part 4 第四部分

准备上线

本部分内容

第 15 章 让网页适应 IE 和其他浏览器

搞定浏览器兼容性能占到网页开发工作量的一大部分，特别是当你面对大量受众的时候。你无法为他们指定一款浏览器来看你的网页，所以你不得不让你的网站能够在尽可能多的用户面前正常显示。你已经可以让 Firefox 浏览器正常显示网站了，现在需要学习如何让它在其他浏览器中看起来没有问题，这里说的其他浏览器包括了 IE。现在你会意识到，那些为了让代码符合网页标准而做出的努力是值得的，因为到目前位置，大部分浏览器都应该可以正确显示你的网站了。

15.1 确定要支持哪些浏览器

在曾经长达 6 年的时间里，网页开发人员只要满足 IE6 就可以了。这个浏览器有一个诡异的呈现引擎，但是 6 年时间在 IT 界算是很长的了，足够让大多数人找到问题在哪里，并且开发出相应的解决方法。随着 IE7 的发布，一些新的呈现显示问题又浮出水面，因为微软对之前的一些问题做了修正，同时还改动了其他地方。在我撰写本书的时候，IE8 也将完工，看上去它也会带来同样的问题。

与此同时，Firefox 和 Safari 经历了各自的三个版本。浏览器跟其他软件一样，都在进步，你无法做到让你的网站在每种浏览器的每一个版本中都能完美显示且功能完好，无论是时间还是精力都不允许你这样做。最终，你不得不决定哪些浏览器是你要支持的。

15.1.1 支持浏览器

如果你够胆大，可以只支持一款浏览器，前提是你的生意允许你这样做。比如说在为公司设计内网系统，如果你可以指定所有人都使用 Firefox，那么就会节省大量的时间。CHAMP Software 是一家在明尼苏达州曼卡托市的公司，为健康护理产业开发网络应用，他们设计的产品就只支持 Firefox 浏览器。

你还可以仿照雅虎，为你的用户推荐一款浏览器，但是这个方法可能会被专业开发人员取笑。在上世纪 90 年代，无数的业余网页上都有“用 IE 获得最佳浏览效果”的字样。在图 15-1 中，你

可以看到雅虎也采取了相同的手段。

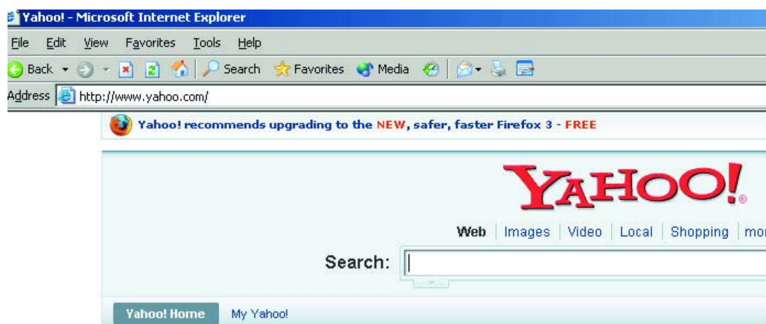


图 15-1　雅虎推荐使用 Firefox 3

不要觉得这是一种偷懒的方法。虽然让网站适合全平台并不是一个多么困难的挑战，但是这个做法不能最有效地利用时间、精力和金钱。至少要估计你做出的决定会导致多少潜在用户的流失，并且将这个流失作为一个考虑的因素。相对于商业网站，企业内部应用通常可以在很大程度上避开这个过程。

15.1.2　只支持某些特性

你可以决定网站的某些功能在某些浏览器中无法使用。微软就提供了两个版本的网页版 Outlook，一个是只支持 IE 的完整版，另外一个则是在其他任何地方都能使用的轻量级版本。完整版本有树形菜单来控制收件箱，外加一些只支持 IE 的特有功能。轻量级版本缺少很多高级功能，但是人们还是可以通过它来查看 E-mail 的。

最终目标是让页面的功能可以满足所有人的基本需求。

15.2　关于浏览器的一些数据

通过一些关于浏览器的数据，可以找到用户们都在用什么样的浏览器。Hitslink.com 维护着一份关于浏览器市场占有率的报告[①]，这份报告提供了一个不错的“市场快照”。同时，W3Counter.com[②]和 StarOwl[③]也会提供比较准确的数据。在做出关于浏览器种类的相关决定时，参考多个来源的数据是很有帮助的，特别是当你在启动一个新网站的时候。另外，网站上线一段时间之后，你应该有一份自己的日志，用来帮助你日后做出判断。

① http://marketshare.hitslink.com/report.aspx?qprid=0。
② http://www.w3counter.com/globalstats.php。
③ http://www.statowl.com/web_browser_market_share.php。

在我撰写本书的时候，多数报告显示 Firefox 占有了 17%～26%的市场份额，而大部分用户还是在使用 IE7，它占到了市场份额的 42%。出人意料的是，有 15%～27%的用户仍然在使用 IE6。我们不得不让自己的网站支持 IE6，因为放弃 20%的潜在用户显然是不明智的。

你的客户和客户的客户多数都是用 IE 的，这个猜测一般不会有错。你开发的网站一定要保证功能正常，并且它的外观在面对各种浏览器的时候能保持一致并且美观。只有这样才算是完美地完成了工作。

15.3 Internet Explore：你无法逃避的恶魔

我认为没有哪个坚持网页标准的网络开发者会喜欢跟 IE 打交道。在本书的讲解中，我已经提到了许多我们需要面对的日常问题，其中包括元素和模式呈现的区别。IE 的很多方式都是错的，但是由于它突出的受欢迎程度，你无法忽视它，而且你的顾客也不会去理会你的个人喜好。

每一个预装微软的 Windows 操作系统的电脑都会随机装有 IE。虽然用户们其实可以随意选择一个其他的浏览器，但是一般用户安装 Firefox、Safari、Opera 或者是 Google Chrome 的几率还是很小。我会把我能接触到的地方全安装上 Firefox，我的家人和朋友也会用 Firefox，因为看中了它的安全性。但是这不能代表大众的情况，因为我是一个技术人员，我的大部分朋友和家人也是。作为开发者，我们必须持有一个概念，就是大众用户不会跟我们用一样的工具。

当心你的数据来源

W3Schools 上有一个总是被引用的浏览器统计数据网页①，页面上列出了各个浏览器的详细市场份额。如果你去查看那个页面，会发现 FireFox 的市场占有率跟 IE6 和 IE7 加起来的占有率差不多。但是，你要记住这个页面的主要用户是技术人员，页面上有如下的免责声明：

“W3School 是一个为那些对网页技术感兴趣的人建立的网站。这些用户会比普通用户更喜欢用非主流的浏览器。一般的用户还是会倾向于使用 IE，因为它是 Windows 自带的浏览器，这让普通用户不需要去找其他的浏览器使用。”

Firefox 是一个很好的浏览器，它的市场占有率的快速增长得益于它的用户和它的投资人。但是当你在收集浏览器相关的统计数据时，特别是这些数据是用来支撑你的决策的时候，注意检查信息来源的可信度。

一些观点

微软 IE 浏览器的这些问题并不是微软恶意使然，虽然这种说法让我们感觉不错。事实上，

① http://www.w3schools.com/browsers/browsers_stats.asp。

微软只是在计划的基础上推出了网页浏览器，它关注更多的是如何让它自己的产品在上面正常运行，像.NET Framework 组件、ActiveX 控件、Outlook 网页版和 ShairPoint 等。IE 不过是微软网络相关产品的配送品而已。

这些产品为微软带来了巨大的利润，所以它必须要在一定时间内支持这些产品。所以微软不能去修复那些我们都知道并且讨厌的问题，因为这会导致它自己的应用崩溃。



小乔爱问……

什么时候可以终止对一种浏览器的支持

这就需要你、你的客户和公司，以现有用户和潜在用户的分析数据为基础来做出类似的决定。看看我们在 15.2 节中提到的那些浏览器的统计数据，截至我写这本书的时候，IE 的活跃用户仍然比 Firefox 要多。如果你只希望网站面对 Firefox 的用户，那么这个数据对你来说关系并不大。但是如果你的目标用户是每一个人，那么你要注意别把大部分潜在用户拒之门外。

我在决定支持哪种浏览器时一直遵守这条准则：支持市场份额最大的两个版本的浏览器，同时保证网站在其他浏览器中可用且可读。不要求在所有浏览中看到的网页完全相同，但至少用户可以正常使用网站。

其实，关于浏览器和功能的选择性支持，会极大地影响客户的收益，以及客户付给你的报酬。

15.4 IE7

IE7 似乎还蛮喜欢我们的网页的。在 IE7 里面，内容居中、PNG 的透明都没有问题，所有的东西也都在它们应该在的地方。究其原因，是因为我们保证了代码是完全符合标准的，于是避免了将 IE 带入它自己的诡异呈现模式。

决定呈现模式

有些时候你无法得知你所处的呈现模式是什么，但是用 JavaScript 可以帮你了解这个信息。在页面的头部加入以下程序片段，看看页面是否是在标准模式下呈现的：

```
<script type="text/javascript" charset="utf-8">
      if(document.compatMode == 'CSS1Compat'){
        alert("Standards mode");
      }else{
        alert("Quirks mode");
      }
</script>
```

15.4.1　IE的诡异模式

在 11.5 节中探讨盒模型的时候，提到过 IE 的“诡异模式”。然而，为了让 IE 能进入标准模式，仅设置一个正确的 doctype 是远远不够的。

15.4.2　XML序言

一些网页编辑器（或者模板）会把 XML 序言（XML prolog）放置在文档的顶端。这种预置的序言会让 IE6 用诡异模式进行呈现。如果你在文档中看到了下面一行代码，你应该将它移除：

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

15.4.3　在doctype上方的注释

开发者喜欢用注释来向其他人解释页面的功用，或者是用注释标明一些稍后其他人可能会用到的信息。然而，如果你将注释放置在了 doctype 声明之上，IE6 和 IE7 都会进入诡异模式。

所以，为了“迎合”IE，千万不要在 doctype 之前放置任何内容，这样它才会以标准模式呈现。

15.5　IE6

用 IE6 打开 Foodbox 网站，你就会发现 Banner 中的图片不透明了，双栏布局也被破坏了，总之很糟糕。即便我们保证它是以标准模式运行的，但 IE6 还是处理不了一些东西。如图 15-2 所示。



图 15-2　IE6 对页面有很大的意见

网页测试

在现如今的网页开发过程中，跨浏览器测试是非常重要的，由此产生了很多种测试方法。以下是我用过的一些方法。

- crossbrowsertesting.com[①]网站提供了不同平台中为不同浏览器设置的镜像，其中平台包括了 Linux、Mac OS 和一些不同版本的 Windows。
- 微软提供了专为开发者准备的虚拟机[②]，虚拟机中可以有不同的操作系统和浏览器。不过那些镜像很快就会过期，你可能需要不时地更新镜像。
- IETester[③]可以让你同时运行数个不同版本的 IE。这个软件只有 Windows 版本。

现在，跨平台跨浏览器的页面测试变得非常容易，然而如果你追求最简单的方式，我向你推荐 Mac 平台。在我的 Macbook Pro 上可以用 Safari 和 Firefox 来开发，然后还能很方便地用本地虚拟机来运行 Windows 和 Linux 测试页面。

15.5.1 修复不正常的地方

有很多 CSS hack 和资源能让你的网页在 IE6 中正常显示，但是用 hack 来解决问题是一种不太好的开发方法，因为 hack 总会被修正的。你需要的是一种更好的方法来定位用户的浏览器，还好微软为我们提供了一个几近完美的解决方案：条件式注释。

你可以用条件式注释来针对所有的 IE，也能制定某种特殊版本的 IE。这些注释只会被 IE 辨认，其他的浏览器会把它们当作一般的 HTML 注释视而不见。

working_with_ie/index.html

```
<!--[if IE 6]>
  <link rel="stylesheet" href="stylesheets/ie6.css"
        type="text/css" media="screen">
<![endif]-->

<!--[if IE 6]>

<style>
   #header img{behavior: url(stylesheets/iepngfix.htc)}
</style>
<![endif]-->
```

① http://crossbrowsertesting.com/。

② http://www.microsoft.com/downloads/details.aspx?FamilyId=21EABB90-958F-4B64-B5F1-73D0A413C8EF\&displaylang=en。

③ http://www.my-debugbar.com/wiki/IETester/HomePage。

这个例子指定浏览器去加载一个额外的样式表。我们将会用这种方式来解决我们遇到的呈现问题。将以上代码添加到原有页面的样式表声明的后面，然后在 stylesheets 文件夹中新建一个叫做 ie6.css 的文件。

15.5.2 解决分栏的问题

现在页面的主区域看起来似乎太宽了一些，无法像它在其他浏览器中一样，容纳在侧边栏旁边的区域内，所以它就跑到了侧边栏的下方。我们遇到了 IE6 的一个 bug，这个 bug 被称作双外边距 bug。当一个有左外边距的元素是向左浮动的时候，IE6 会将这个外边距值加倍。这个 bug 对拥有右外边距的右浮动元素也是有影响的。

最简单的修复方法是为受影响的元素加上 `display:inline` 属性。解决这个问题的实际过程，其实是对受影响的元素准确定位的过程。

通过检查后发现，引起双外边距 bug 的真正元凶其实是 `main_text` 区域。在为 IE6 准备的样式表里加上下面的代码，问题就能解决了：

working_with_ie/stylesheets/ie6.css

```
#main_text{display:inline;}
```

15.5.3 修正透明问题

IE6 不支持 PNG 的 alpha 透明值，但是网络上有成千上万种的解决方法。当然了，这其中没有一个方法是特别简单的，而我最喜欢的一个方法叫做 TwinHelix[①]。

从那个公司的页面上下载修改包，解压，然后把 iepngfic.htc 文件和 blank.gif 文件放入 stylesheets 文件夹中。

打开 iepngfix.htc 文件，找到这行代码：

```
if (typeof blankImg == 'undefined') var blankImg = 'blank.gif';
```

然后将它改为：

```
if (typeof blankImg == 'undefined') var blankImg = 'stylesheets/blank.gif';
```

为了正常运行，HTC 文件会用一个 blank.gif 文件来完成透明的部分，而这个文件需要的 blank.gif 的链接是你提供的，链接地址应该是 blank.gif 相对于应用了这个方法的 HTML 文件位置的地址，而不是相对于样式表的地址。

① http://www.twinhelix.com/css/iepngfix/。

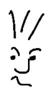
为 index.html 文件的条件式注释加入以下代码：

working_with_ie/index.html

```
<!--[if IE 6]>

<style>
   #header img{behavior: url(stylesheets/iepngfix.htc)}
</style>
<![endif]-->
```

这样会载入那个特殊的 CSS 动作，兼容 IE6。



小乔爱问……

为什么我们要手动修复这个问题，用已经成型的 IE7-js 不是更好吗

很多开发者将类似于 IE7-js[①]的项目视为简单的修复手段。但是作为开发人员，你应该对页面中的代码负责。如果你能完全搞明白像 IE7-js 这种项目中每一行代码都是做什么的，为什么它能让页面兼容这些浏览器，那么你才可以没有顾虑地将这些东西加入到页面之中。但是你应该注意到像 IE7-js 这种方案是很理想化的方案，它试图将所有的情况都考虑在内。我不喜欢在我自己的项目中加入额外的代码，我只需要能解决我问题的代码。我宁愿只解决我的问题，而不是用一个大而全的库。这个大而全的库可能还会跟我的其他东西有兼容性问题。

库和框架本身都是很好的，但是你需要对项目中用到的所有东西都完全掌握，这一点很重要。当出现问题的时候，不管那段代码是不是你写的，最终负责的都会是你，你需要解决出现的任何问题。

15.5.4　修复页头图片下面的空白

页头图片有一点点内边距，将页面的整个内容向下推了一点，于是页面就没有如我们所期盼的一样和背景对齐。在标准模式中，图像元素是内联元素，它们本来应该和其他内容一样跟标准线对齐，在自己的上方留出一点空间给下行字母。看上去是 IE6 遵守了这个标准，而其他的浏览器在这里则引入了一个例外——这种模式也叫作“准严格”模式。快速的解决方法是将图像的显示方式变成 block 方式：

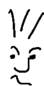
working_with_ie/stylesheets/ie6.css

```
#header img{
  display:block;
}
```

① http://code.google.com/p/ie7-js/。

这个修复不难，现在将所有东西都放回原位了。虽然本可以将这些改动从 IE6 样式表中移走，加入到主要的布局样式中去，但是我们只在 IE6 中发现了这个问题，所以这个步骤并不是必须的。

对于 IE6 的修改到这里就差不多了，基本上所有的东西都在它们应该在的位置。现在你可以回忆一下刚刚的工作量，然后再来权衡一下要让网站支持 IE6 是否值得：下一个项目可能更加复杂，需要更多的修复工作，而目标用户却可能不太一样。但是，即便下个项目同样需要支持 IE6，你对那些常见的问题和解决方案也应该比较熟悉了。



小乔爱问……

这方法对多个页面有效吗

如果网站和应用中有多个页面的时候，HTC 文件的相对路径会随着页面而变化，它的地址取决于 HTML 文件在页面结构中的位置。这种情况下，你就需要为 HTC 设置一个相对于根目录的链接。这样做了之后，可能在本地是看不出来效果的，要等部署到服务器上才能查看效果。

在这个例子里，文件不会部署到服务器上，所以将 HTC 文件中连接到 blank.gif 的句子改成：

```
if (typeof blankImg == 'undefined')
   var blankImg = '/stylesheets/blank.gif';
```

然后将 HTML 中调用 HTC 文件的语句移动到 ie6.css 中，注意修改文件链接路径。

你也可以不用透明图片，但是，那样显然就不怎么美观了。

15.6 IE8

现在，微软对那些希望在“标准环境”下写代码的开发者也是越来越友好了。IE8 开始允许用户使用特定的呈现模式。IE8 提供了一些可供选择的模式，其中包括 IE5 和 IE7。根据微软的说法，IE8 呈现模式提供了对行业标准的良好支持，包括 W3C CSS Level 2.1 规则、W3C 选择符 API 和对 W3C CSS Level 3 规则（起草中）的有限支持。

这听上去很美，但是其中有一个很大的缺陷。IE8 的呈现模式并不识别 doctype，但其他浏览器还是依赖 doctype 确定呈现模式。还好，微软在 IE8 中还继承了一个功能叫做 IE8 仿真[1]，这个

① http://msdn.microsoft.com/en-us/library/cc288325(VS.85).aspx#。

仿真功能会重视 doctype，用标准模式进行呈现，并且在将进入诡异模式的时候自动切换到 IE5 模式。你应该看一看下面 IE8 的兼容性数值列表。[①]

值	描　述
IE=8	网页将支持IE8模式，也称作IE8标准模式
IE=7	网页将支持IE7模式，也称作IE7标准模式
IE=EmulateIE8	如果网页指定了基于标准的DOCTYPE类型，那么网页支持IE8模式；否则会进入IE5模式（诡异模式）
IE=EmulateIE7	如果网页指定了基于标准的DOCTYPE类型，那么网页支持IE7模式；否则会进入IE5模式（诡异模式）
IE=Edge	网页会兼容浏览器所支持的IE最高版本，这个值通常会在测试的时候被用到

为了让 IE8 能够正常显示 Foodbox 网页，需要在 HTML 文档中页头的 head 标签之后加上如下代码：

working_with_ie/index.html

```
<!--[if IE 8]>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8" >
<![endif]-->
```

注意，需要将这一段注释放在文档的顶端，紧接着 head 元素的打开标签之后。你应该尽早写入关于兼容性的代码，至少要早于任何样式表和 JavaScript 文件的引入语句，这样 CSS 和脚本文件才能被正确解析。一旦你设置了这个修正，IE8 就可以很顺利地工作了，它甚至会支持在打印样式表中用到的高级 CSS 功能。

这个方法的问题

在 IE8 中微软终于开始支持其他浏览器已经支持了多年的标准，但是它并没有把这种支持设置成默认值，所以为了网站能够全部正确显示，你不得不在每一个页面上都加上那段代码。但是你加上的是一个 meta 标签，所以至少这种方法看起来比用 CSS 或者 JavaScript 来解决问题要高级一些。我个人对这个实现方法并不是很热衷，但是相比较花上数小时来钻研脚本，我还是会使用这个只用一行代码就能解决问题的方法。

15.7 其他浏览器

将网页放在其他浏览器中看看效果也不会有什么损失。令人开心的是，Foodbox 在 Mac 的 Safari 表现良好，如图 15-3 所示。同样，对于采用了兼容标准的呈现引擎的 Google Chrome 浏览

① 参见 http://msdn.microsoft.com/en-us/library/ms533876(VS.85).aspx。

器来说，也不用对页面做出任何修改（参见图 15-4）。

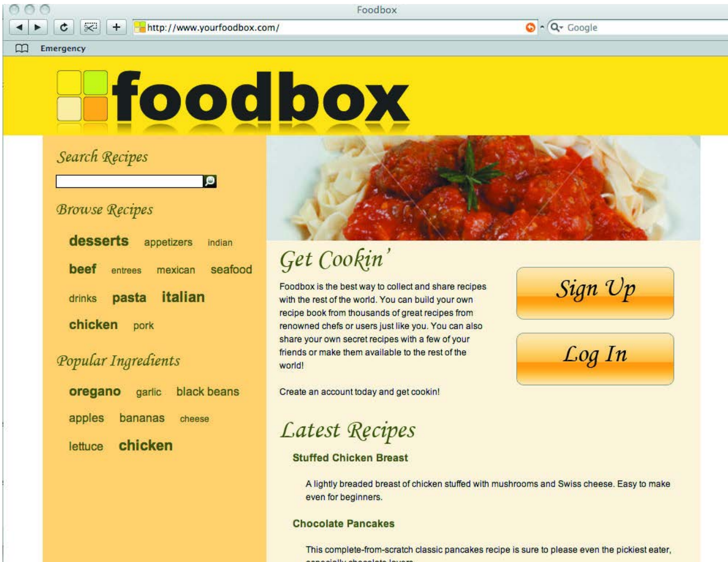


图 15-3　Foodbox 页面在 Safari 中看着也很好

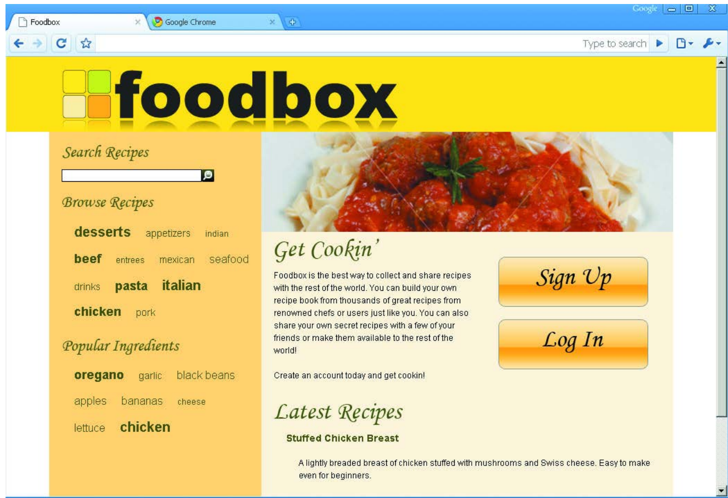


图 15-4　Foodbox 网页在 Google Chrome 中看起来很不错

我要指出的是，这不是因为我们特别幸运，而是我们遵照标准写出一个合法文档后理应得到的好结果。由此我需要指出，不要用 IE 进行开发，那完全是浪费时间。你要做的是在一个与标准兼容的浏览器中做开发，然后针对其他怪异的浏览器做出修正。

15.8 小结

为了能让网页被更为广泛的用户所接受，应该进行跨浏览器的开发。很让我开心的是，这种开发正变得越来越容易。如果你遵守了标准，作品就会运行正常。现在 IE6 正在退出舞台，IE7 正在被 IE8 取代，各个浏览器之间的差异正在慢慢缩小。这对于开发者和用户来讲都是一个好消息。

第 16 章

可访问性和可用性

你需要考虑用户的多样性。比如说，一个色盲用户能否正常使用你的网站？盲人呢？或者那些无法正常使用鼠标的用户，他们怎么办？标签云中链接的间隔是否因为太小而导致协调运动技能有缺陷的用户无法使用？

另外，那些网速很慢的用户怎么办？对他们而言，页面的载入速度是否够快？还有，在移动设备上，比如手机，你的页面表现如何？

对于程序员来讲，这些关于*可访问性*和*可用性*的概念也许会显得比较陌生。这些话题也是最近才开始流行的。在本章中，你会了解面对不同用户时遇到的不同问题，以及如何改进网站让它变得人人可用。

16.1 可访问性对你来说意味着什么

跟网页开发人员聊天的时候，我总喜欢问他们这个问题：听到*可访问性*这个词，你会想到什么？那些各种各样的答案通常都很有意思。有些人完全不知道还有这样一个概念，有些人认为这个东西是专为残疾人准备的——可访问性包含的内容远不止于此，它其实泛指跟“页面访问”有关的问题。

当评价一个应用或者网页是否具有可访问性的时候，是说这个东西可以被任何用户以任何互动方式查看或者使用。这包括了跟残疾人相关的辅助技术，但同时也跟那些用旧式电脑、低速互联网、移动电话、PDA 或者游戏终端的用户有关。这并不是说页面需要在所有的平台上都保持一致，但至少它能够在这些地方帮助用户达到其想达成的目标——无论用户是想查找信息、阅读文档或者是购物。

如果网站只在开启JavaScript的时候才可以正常使用，那么它就不具备可访问性。因为并不是所有浏览器都支持JavaScript，也不是所有用户都会开启它，而且，并不是所有浏览器都会用同一种方式解析JavaScript。

如果网站必须在用户安装了 Flash 之后才能正常工作，即便它的菜单动画绚丽无比，它也不具备可访问性。如果因为客户的 iPhone 上没有安装 Flash 而无法单击菜单项，那最终你可能会失去那部分客户。

如果网站上的图片太多导致那些网速很慢的用户无法使用（是的，还有这样的用户），那么网站在可访问性上也存在问题。

如果你跟大多数程序员没两样，第一反应可能就是必须要用像 Flash 和 JavaScript 这样的技术来让网站更吸引人、可用性更佳并且更有竞争力。这个理由是很恰当的。像 Gmail，如果少了对 Ajax 的支持，它会变得非常难用。但是即便会功能不全，在没有 Ajax 的情况下，Gmail 还是可用的。Google 把这个不完整的 Gmail 做成了一个可以正常使用的东西，虽然它比常规版要笨重很多。

网站应该是人人可读并且人人可用的。它可以没有太绚丽的外表或者多么流畅的使用感，但是它应该能够让用户达到他们的目的。

16.2 关于可访问性的基础问题

这一节涵盖了关于可访问性的基础问题。

16.2.1 盲人

双目失明的人通常需要使用*屏幕阅读器*，特别是那种可以通过页面上的文字自动合成声音的软件。市面上有好几种屏幕阅读软件包，包括比较受欢迎的 JAWS。考虑到软件各自支持的特性，它们各有各的优缺点。

1．屏幕录像、视频和页面导览

几乎所有新网站都有类似于页面导览的东西，要么是一些截图，要么是一段视频简介。当设计一个导览或者屏幕录像的时候，要考虑到这些东西对于盲人用户的使用感受。当你为屏幕录像设置语音辅助的时候，注意语言风格，要让解说听起来跟转播比赛一样，要向观众描述正在发生的事情——不要只说“单击这里”，而要描述“选择‘新的菜谱’这个链接”。即便是少量的描述，对盲人甚至是有视力障碍的人来说，都是有重要意义的。

同样地，你也应该对图片形式的导览做同样的处理，在图注中用描述性的语言做标注。这会增强用户使用时的愉悦感。

2．颜色

如果用户是盲人，那他们自然也不会对页面上的颜色有反应。之前我们花了一整章的内容

讨论如何利用不同的颜色来调动用户的情感。遗憾的是，这些原则都不适用于盲人用户。因此，你要利用其他方法来向他们传递一些重要信息。比如，以前可能会用不同的颜色来标注表单提交后的错误信息，现在你可能要将错误信息的表述列在表单顶部了。这些信息可能需要用 strong 或者 em 标签标注出来。很多屏幕阅读器都会把被这些标记包括的内容读出来，以引起用户注意。

3. alt属性

9.4.7 节的“默认文字”框注中，我们曾经对 alt 属性有过一些讨论。alt 可能是最广为人知的增强页面可访问性的途径。然而，如果使用有误，它也会给屏幕阅读器用户带来很多烦恼。

alt 属性的作用是给图片加上描述性信息。但是很多网页开发者当作一个任务般的事情来处理，所以他们时常会随意地写一些信息，甚至有时候只把文件名再打一遍——后者更加要不得。

大部分屏幕阅读器用户都不需要听到诸如“公告”、“对钩”或者“一个穿着红毛衣的女士”的信息。

如果页面上的那张图片只是为了装饰而设计的，那么它的 alt 属性留成空白就可以了：

```
<img src="/images/circle.gif" alt="">
```

更好的方式其实是将这些美化页面的元素放入样式表里面，只在 HTML 文档中留下跟内容相关的图片。譬如可以用下面的方式来实现图片样式的列表：

```
ul{
  list-style-image: url("/images/circle.gif");
}
```

用户真正想听到的内容，可能是图片中的文字，或者是图表中的“产品生产地”信息。alt 里要么是跟内容相关的描述信息，要么就留为空白。但是一定要写上 alt 属性，大多数屏幕阅读器都会自动忽略空白的 alt 属性。

最后，写 alt 属性的时候，不要用“这是一张某某图片”这种文字开头。屏幕阅读器在读 alt 属性时会告诉用户它遇到了一张图片，所以用户其实已经知道他将要听到的内容是一张图片了。

4. 图表

对于盲人来说，图表类图片几乎是没有意义的，除非你能为这些图表提供一些描述。除

了之前提到的 alt 属性之外，img 标签还提供了另外一种叫做 longdesc 的属性。这个属性可以用来描述图表。大多数情况下添加图标描述都是有益于用户的，特别是描述非常详细的时候。

5．拼写和语法

页面上的文字是可读的，屏幕阅读器通常能很好地运行。在填充页面内容的时候，一定要特别注意保证文字意思明确、书写正确以及语法无误。语法和句子结构很重要，因为屏幕阅读器会读出它扫描到的每个字。想想那些同字不同音的例子。像“重音”这个单独的词，到底应该是重量的重，还是重复的重？你肯定弄不清。同样地，如果你在写句子的时候不注意，屏幕阅读器也弄不清。屏幕阅读器会尽量通过上下文的联系紧密程度来做出判断，但是它依赖的是文章中的正确书写和语法。

把每个字都写正确也非常重要。想象一下如果你漏写了一个单词中的一个字母，那会对屏幕阅读器造成多大的困扰啊！

最后，标点很重要。注意单引号、逗号以及其他标点的运用。面对不同的标点，屏幕阅读器也许会做出不同的停顿处理。比如当遇到逗号的时候，屏幕阅读器可能会停顿一下，以帮助用户理解内容。

我通常会先用电脑检查一遍自己的语法和拼写，然后再找一个人帮我校对。如果有其他人帮你写内容更好，但是一定要自己做一遍校对。即便是客户给你的文本，如果放在网上之后发现了打印、语法和书写错误，最后挨骂的还是你。

我的校对规则

我写过很多东西，从内部文档到博客以及这本书，都是我的写作范畴。当然，在写作的时候我经常犯错，为此，我形成了一套自己的校对措施。在持续使用的情况下，我的这个措施还是非常有效的。

首先，大声朗读你写的东西，如果能够不发笑地顺利读下来，那么第一步算是成功了。

其次把句子倒着读一遍，这样除去语境的阅读可以帮助你发现重复字词和书写错误。

最后，找另外一个人来帮你读一遍，这样你就能听听自己写的东西了。这一步很重要，它还可以获取他人对你文章的反馈。

6．JavaScript和Ajax

JavaScript 和 Ajax 技术让程序员有能力去创造更丰富的用户界面，比如说可以做到让用户在一个页面上编辑内容，而不是跳转到一个表单，你还能根据用户输入的内容来显示和隐藏某些页面。另外，这两个技术还能在页面内弹出对话框加载页面，就像 Lightbox[①]一样，从而淘汰掉老旧的浏览器提示框。

这两个技术的问题在于，虽然大多数屏幕阅读器声称支持 JavaScript 和 Ajax，但实际上很少能有屏幕阅读器可以完美地支持它们。在使用了这两个技术的页面上，屏幕阅读器很难分辨哪一部分页面发生了变化。当你开发页面的时候应该保证所有的基本功能在没有 JavaScript 的时候也可以正常运行，那些基于 JavaScript 的华丽特点，则是根据需求而加上的对功能的补充。这样一来你不但能建造一个“进可攻退可守”的网站，而且不会在可访问性出差错之后才去亡羊补牢。

比如说要做一个 Ajax 的表单，先实现一个常规的表单提交过程，然后再不唐突地做一些改动，让它具有 Ajax 的功能。接下来在服务器端，用代码决定是响应常规请求还是 Ajax 请求。前者可能会在一个新页面中打开表单，而 Ajax 请求则会返回一个 JavaScript，回应已经改变了原有页面的形式和内容。这个实现过程并不会消耗太长的时间，却为更多的用户提供了良好的使用体验。

你应该用不同的屏幕阅读器来测试页面，看看它们是怎么处理网站的。同时，那些生成屏幕阅读器的公司也在不断努力，为了让网络变得更友好。

如果在建站的时候能够一直关注这些问题，那么对于用户来讲，你创造的东西会更加友好。为图片加上详尽的描述能帮助用户更好地理解段落内容。检查语法和书写则是你必须要做的工作，而用户界面的“去功能化”则能让那些没有用复杂浏览器的用户使用你的网页。有时候我会用手机浏览互联网，但是有几个我很喜欢的 Web2.0 网站却不支持手机访问。

7．Google也看不见

Google——还有其他搜索引擎——也是“盲人”，它们会跟屏幕阅读器一样，去分析你的页面。一些特定的内容，像图片、视频、图表和 Flash，也是不能被分析的。有时，页面结构也会妨碍用户使用。

我向你推荐基于文字的浏览器，用它们来检查页面的可访问性，比如说你可以用 Lynx（参见图 16-1）。或者你还可以尝试着将浏览器里的 CSS、JavaScript 和图片去掉，然后浏览这个网页。

---

① http://www.huddletogether.com/projects/lightbox2/。

这样的方式可以让你对网站的可访问性有个大概的了解，同时你还可以大概知道在搜索引擎眼中网站是什么样子的。

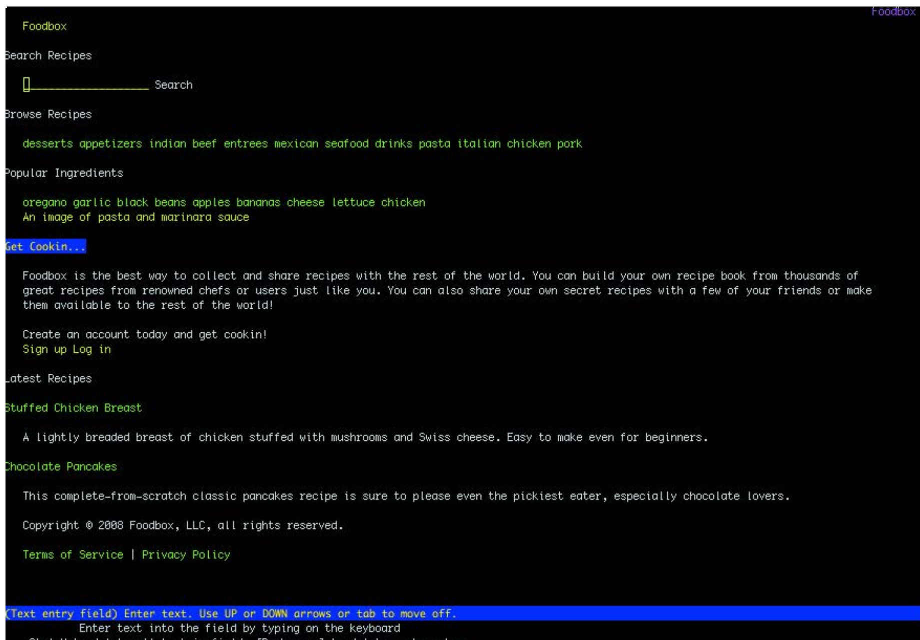


图 16-1　在 Lynx 中的 Foodbox

16.2.2　色盲用户

如果你在页面上用颜色传递重要信息，色盲用户可能会遇到问题。色盲用户通常不能区分某两种颜色，比如说无法分辨绿和红，或分不清红和黑。

因此，在设计图表和页面上其他区域的时候，要注意颜色的搭配具有足够的对比度。如果你在一个黑色的背景上写上了红色的字，那么有些色盲用户可能完全无法辨认出来。

1. 红绿色盲

红绿色盲是最常见的两种色盲。有这种色盲症的人群无法区分红色和绿色。

在图 16-2 中你能看到红色盲眼中的 Foodbox，注意看肉丸上的红色酱料，它们在红色盲眼中是深灰色，而整个页面的颜色则偏棕色。也就是说对于这群用户，红色其实是某种很暗的颜色。所以在为页面选择配色的时候要注意到这一点。

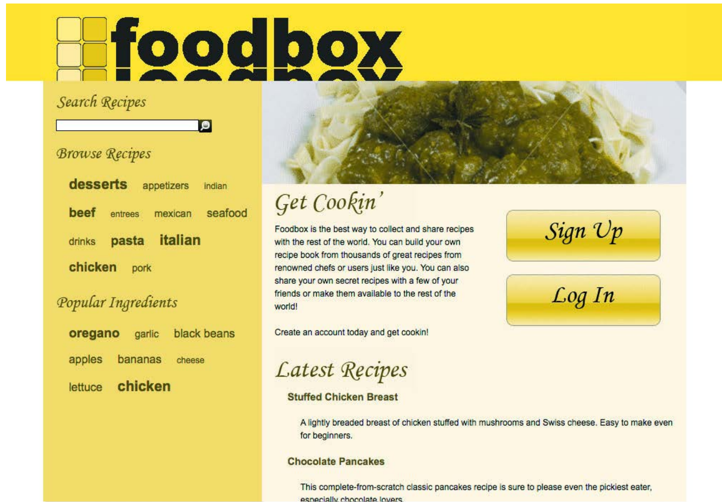


图 16-2 在红色色盲用户眼中的 Foodbox（另见彩插）

绿色盲症的情况跟红色盲差不多，只是对于这种患者而言，红色和绿色没有什么分别。在图 16-3 中你可以看到他们眼中的 Foodbox 也很与众不同。



图 16-3 绿色盲的用户眼中的 Foodbox（另见彩插）

2. 蓝色盲

蓝色盲很罕见，这种色盲的症状是无法分清蓝色和黄色。在这类人群眼中，Foodbox 的网页看起来偏粉（参见图 16-4）。



图 16-4 蓝色盲用户眼中的 Foodbox（另见彩插）

16.2.3 有视觉缺陷的人

有视觉缺陷的人遇到的问题可能不会那么明显和直接，他们通常会用像 ZoomText 或是 OS X 自带的放大软件。所以你并不用太担心字体的大小——当然用太小的字体肯定不行。

但是，放大镜也是有缺陷的。视觉有缺陷的用户通常会把注意力集中在他们放大的区域，而忽略屏幕上的其他内容——就像穿过一个纸筒看屏幕一样，你不得不将纸筒移来移去才能看到所有的东西。所以会有些内容是设计师需要你看到却被你忽视了的。从开发者的角度来讲，无法控制人们使用可访问性工具。那么该怎么解决这个问题呢？你应该结构化页面，达到让用户扔掉放大工具的目的。

通常视觉缺陷用户并不喜欢那些单独为他们开发的页面，就像其他残疾人一样，他们希望被平等地对待。如果开发者没有为这些单独的页面做持续的更新，他们会更郁闷。我碰过几次类似的事情，搞得很不愉快。

我平日会经常使用辅助性工具，如放大工具，而我的建议是让用户自己去操心工具的使用问题，开发者则应该想办法将重要的信息放在一起，这才是你们应该操心的地方。那些需要放大工具的用户都有自己的软件，在网络应用中，IE7、FireFox 和 Opera 都有让用户放大整个页面的工具。而 PC 或者 Mac 中一些全屏放大工具可以将整个使用空间放大，而不单单针对浏览器。

16.2.4 有听力缺陷的用户

网络也还算是视觉媒介，越来越多的网站开始用视频来展示页面功能，或者是共享信息。诚然，听力障碍用户是可以看视频的，但若你没有为视频或屏幕录像配上解说文字、标题或者是字幕，那么这些用户肯定会有种被遗忘感。如果你在经营一个播客，请记住在广播的同时，为那些

听力有障碍的用户提供一份可以点击的解说图。

老人、退伍军人、婴儿潮时代[①]出生的用户可能出现听力障碍，现在的年轻人中有听力障碍的人的百分比也在极速上升[②]，所以你不能忽略这些用户。如果你在页面上提供了有声视频，记得声轨的音量要大，并且在小音量的时候，声音质量也要清晰——这样做可以保证用户能够调整到适合他们的音量大小。

16.2.5　行动障碍和没有鼠标的用户

我们的服务对象或者产品的使用者，他们的设备不一定就和我们自己使用的设备是一样的，而作为计算机的日常使用者，我们时常会忘记这一点。譬如有的用户因为手部残疾或者肌肉萎缩而无法操作鼠标，他们可能会使用鼠标的替代品。另外，盲人用户也不是用鼠标浏览的。

那么这些用户如何浏览网页呢？有些人会用一种特殊的管子，当他们吸气或者吹气的时候鼠标指针就会移动；还有一些用户只能用键盘。另外还有那些讨厌鼠标只喜欢用快捷键的程序员们。

那些使用键盘的用户依靠的是快捷键和 Tab 键，你需要试试他们的浏览习惯，只用键盘，然后记录下遇到的问题。你是否用了滑块控件？那就要记住为它加上一个可以让用户输入的区域。另外，你是否用到了那种需要鼠标单击才能激活页面输入区域？总之底线很明确，不要让你的网页完全依靠鼠标操作。

最后，记得不要用“单击”这个字眼，比如说不要用“单击这里”。因为这个词暗示着用户是使用鼠标的，但是并不是所有人都用鼠标。如果你有一个“单击查看更多”的链接，不妨把它改成“更多信息”这样的链接会更好一些。如果网页的可用性做得好，用户会自然而然地知道如何去操作以获得“更多信息”。其实我个人认为，人们之所以会使用“单击”这个词，是因为看到其他人都在用，我在这里呼吁大家打破这种常规。

16.3　包容一切

当我们说一个网站可访问的时候，指的是它能被所有的用户使用。如果在页面上使用了太多的 Flash 和 JavaScript，那么应该考虑到那些没有这些功能的用户，他们如何同页面互动。当然，这并不意味着你要在建造新页面的时候畏首畏脚，只考虑最通用的标准——毕竟让网站有竞争力也是很重要的。但是你应该尽量为更多的用户提供访问途径。

在我写这本书的时候，Hulu[③]用的是 Flash 来播放我爱看的剧集视频，这个网站也是我最喜爱

---

① 婴儿潮（Baby Boom）主要指美国第二次世界大战后的一段时期（1946～1964），这 18 年间出生人口达到 7 800 万人。——编者注

② MiracleEar 是一家助听公司，它的报告称“有 15%的大学毕业生的听力衰退程度跟他们父母的相当或更严重”。

③ http://www.hulu.com/。

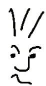
的几个网站之一。但是我不能在我的 iPod 上看这些视频，因为 iPod 不支持 Flash。另一方面，YouTube 虽然也用的是 Flash，但是在 iPhone 发布之后，人们也可以在 iPod 或者 iPhone 上看它的视频了。虽然选择 Flash 对于 Hulu 来说是一个不错的决定，毕竟大多数用户的设备上都是 Flash，而非 QuickTime 或者是 Windows Media。但是如果 YouTube 可以让它的视频在非 Flash 设备上正常播放，那么 Hulu 应该也能做到[①]。

当然，Hulu 并没有承诺说可以在 iPod 上看视频，我只是用这个例子来说明一个现象，那就是你的一个关于技术的决定是如何将一部分用户拒之门外的。所以当你决定要实现一个比较新颖的技术的时候，一定要考虑这个决定对不同的用户群会产生怎样的影响。

16.3.1 导航

在不知情的情况下限制用户浏览网页的方式，是终结一个网站生命的有效方法。导航异常重要。如果导航是一个下拉菜单，那么请保证用别的方法也能取得菜单中的入口。因为用老式浏览器或者是屏幕阅读器的用户是无法使用下拉菜单的，他们只能看到顶级目录的那个选项。为了解决这个问题，你需要制作一个“着陆页”，这个页面会包含下拉菜单里的所有链接。起初，这个方法似乎没有那么完美，但是它除了带来可访问性方面的好处之外，还会为市场部的同事提供一个额外的机会，让他们能够添加一些有质量的关键字和内容。

我还见过无数使用 Flash 影片做系统菜单的网站。虽说很多人的家用电脑里装有 Flash，这种方法还是会给那些使用屏幕阅读器、手机或者 PDA 上的浏览器的用户带来使用问题。不应让用户为了浏览你的网站而去下载安装额外的软件。我的意思也不是说不能用 Flash 做导航，但是你需要为特定的用户提供一种“绕行”的方式。



小乔爱问……

其实 Flash 也是具有可访问性的，对吧

如果由 Adobe 来回答这个问题，那答案会是肯定的。在 Adobe 收购 Marcromedia 的时候，它着实为可访问性操了心，而后者也一直在让它产品的可访问性变得更好。问题在于，即便 Flash 视频本身是有可访问性的，它的具体实现也依赖于那个发布视频的人。即便发布视频的人做到了这一点，它对屏幕阅读器也仍旧不是那么友好。去下一份 JAWS 或者是 Window-Eye 的试用版，看看它们在页面上表现如何。这其实也是老生常谈了：测试，测试，再测试。

① 当然，这里也可能牵扯到了某些法律条款的问题。

16.3.2　处理出错信息

如何在网页上显示错误信息？如果你准备使用弹出窗口，那你要考虑到有行动障碍的用户关闭这些窗口的难度会很大。如果你用另外一种颜色（比如红色）来表示错误信息，那色盲用户可能根本就不会注意到它。另外，如果你用 Ajax 做表单认证，有些屏幕阅读器也不会加以识别。那到底我们应该怎么办呢？

Ruby on Rails 里内置的 scaffold 提供了一个很好的处理表单验证的例子。一旦用户输入了不合法的数据，表单会被刷新显示，页面上会多出一块显示错误信息的地方。信息中还包括了对已出现问题的简短描述。另外 scaffold 还会用红色标记出表单中出错的项目，并为它们加上粗边框。

这个方法还有改进的余地，比如说将错误信息显示在错误项的附近。这也不失为一个很好的方法，因为它没有单单依靠颜色的变化来显示有问题出现。

16.3.3　跨浏览器测试

在第 15 章中，我们讨论了跨浏览器的相关问题，在那一章里针对 IE 和其他浏览器对 Foodbox 作了一些调整。类似地，我们将在第 19 章中讨论如何应付移动设备。除了一句“跨浏览器是可访问性的一个重要组成之外”，其他的似乎也没有可以多说的了。记住你的最终目标——让尽可能多的人可以成为你的用户。

16.4　重要的商业问题

网站建设是一个竞争异常激烈的市场，这可能会影响到你的网站的可访问性。如果你的愿望是建造一个引人入胜、技术领先、创新且有商业魅力的网站，那么就不得不面对这样一个局面，那些你想拥抱的技术并不是人人可用的。你需要与人竞争，同时还需要保证页面能够被市面上 90%的用户无障碍使用。

不管怎样，你都需要从一开始就在脑中埋下可访问性的概念。我能理解网站上线赢取用户对于你的客户来说有多么重要，但如果你只是单纯地建站，想着可访问性的东西可以日后再加，那么你日后可能就会有个大“惊喜”喽。一旦你的用户达到了一定的数量级，你的精力就会集中在添加新功能和解决旧问题上面。可访问性就像“测试驱动开发”：如过你没有从一开始就采用这种方式，那么你永远也不可能采用它了——因为到最后你会变得讨厌它。如果你不能以“这是正确的事情”来说服老板们去考虑可访问性，那么就把他们最能理解的事情放上台面——钱。跟他

们讲如果不考虑可访问性的问题，会流失多少目标市场，具体数字会超过老板们的想象。

如果你是为政府、大学或者公立学校建站，那你还需要去了解相关法规，要了解对于这样的客户，技术方面的交易应该怎样进行。相关法案规定，有些政府机构不得购买不符合 508 条款[①]的技术产品。如果你的目标市场包括这类机构，你则需要保证自己的产品是符合条件并可以向他们出售的。



小乔爱问……

如果可访问性是如此重要，那为什么我们到现在才开始做相关的实现呢

这个问题不错。之所以现在才谈论到这个话题，是因为我有太多的内容需要写了。另外一个原因是这本书并不是专门讲可用性的。但是，如果你往回翻翻书，应该能注意到我一直都有提关于可访问性的概念，包括默认文字、验证、语义化的标签、读写障碍和一些配色的问题。在 Foodbox 上实现可访问性的功能会相对容易一些，因为在设计的时候就已经考虑到这些问题了。在建造下一个网页的时候，你应该从最初的开发开始，就将这一章所提到所有知识考虑在内。

去 http://www.section508.gov/了解更多关于这方面的指南，然后看看它对你有哪些影响。按照这个网站上的指南，你应该可以建造可访问性非常好的网站。实际上我个人认为，关于残疾用户的可访问性，这个网站上的指导要比 W3C 的指导好很多。

16.5 改进 Foodbox 网站的可访问性

从文字浏览器中看 Foodbox 会发现这样一个问题：不管是使用屏幕阅读器的盲人用户，还是无法加载样式表的手机用户，都需要往下拉动相当一部分页面才能登录，或是阅读页面中的内容。搜索框和标签云横在用户和内容之间。

对付这种恼人的小问题，可以新建一些跳转链接用以导航，并将这些链接在主样式表和打印样式表里隐藏起来，这样一来屏幕阅读器会识别出来这些链接，而普通用户无法看到它们。

16.5.1 添加跳转链接

开打 index.html，就在紧接着 Banner 图像的地方加上下面的粗体代码。

① 这里指的是 1998 年美国议会通过的对康复法案第 508 节做出的修改，内容包括 IT 技术产品需要让残疾人可以正常使用。——译者注

accessibility/index.html

```
<div id="header"> <!-- start of header -->
  <img src="images/banner.png" alt="Foodbox">
  <ul id="skiplinks" class="noprint">
    <li><a href="#main_text" accesskey="0">Skip to Content</a>
  </ul>
</div> <!-- end of header -->
```

Skipe to Content 链接指向页面的主文字区域。之前给所有的区域都加上了 id，这对提供具有可访问性的导航是非常有利的。

accesskey属性

在上面提到的链接中，你可能会注意到一个新的属性——accesskey 属性。

这个属性可以让你的用户使用快捷键激活链接、按钮和表单域。这对使用屏幕阅读器的用户是个很大的便利，同时对那些不能或者不愿意用鼠标的用户也是一个很方便的功能。当然你需要以某种方式告知用户这些快捷键的存在，因为他们不可能去看你的源代码来知晓快捷键。

初看之下，我在这里选择 0 作为快捷键似乎毫无道理。之所以添加这个键值，是因为我跟 Twitter 的开发者有一样的认识——这个快捷键对手机用户非常有用。瞧，可访问性并不只是关于残疾人用户的吧。

16.5.2 屏幕阅读器和display:none

在第 14 章中，你学会了如何在相应的 CSS 中使用 display:none 来隐藏某些区域或是元素。有些屏幕阅读器已经开始对这个规则区别对待了。很多关于可访问性的文章都主张使用 display:none 来隐藏跳转链接，但是这种方法似乎已经行不通了。其实可以用另外一个技巧来将这个链接放在页面之外。

快捷键是如何工作的

浏览器如何响应你按下的快捷键是由这个快捷键制定的元素决定的。如果你为一个链接设定了一个快捷键，那么当用户按下这个快捷键的时候链接会被激活。[①]当你将快捷键指向一个表单域的标签的时候，用户的鼠标指针会指向并激活那个表单域。

快捷键基本上可以取代单击鼠标的作用，而且它还可以为用户的工作流加速。

① 在 IE 里，浏览器只是指向这个链接，并不会激活。

16.5.3 用“负位置”隐藏跳转链接

隐藏内容的 CSS 只需要几行就能写好。这里首先要用到绝对定位，这样就可以在 CSS 里面使用元素的 X 坐标和 Y 坐标。在激活绝对定位后，可以把元素的坐标设为−9999 像素，这意味着该元素会被放置在浏览器距左侧边缘的左边 9999 的像素处。[①]

将以下的 CSS 规则添加到 stylesheets/layout.css 文件中：

accessibility/stylesheets/layout.css

```
#skiplinks{
  position:absolute;
  left:-9999px;
}
```

给需要跳转的链接加上 `class="noprint"`属性，这样会自动隐藏打印样式表。

16.5.4 表单的标签

利用页面的快捷键导航，可以进一步帮助移动用户、盲人用户和有运动障碍的用户。为了达到这个目的，只需对表单做一些微调，给它加上一些标签。`Label` 域可以将文字标签和表单域绑定，这样屏幕阅读器就可以将表单域中的值和名字对应起来了。

标签的力量

标签可以提高页面对所有用户的可访问性。比方说，如果你给一个单选按钮或是一个多选框加上了标签，那么用户就能通过选择文字标签来选择这个多选框或者是单击那个按钮。这个方法为那些不能准确使用鼠标的用户提供了方便。

我们可以通过指向一个元素的 ID 来为它指定标签，就像这样：

```
<input type="checkbox" value="yes" id="user_active"
       name="user[active]" />
<label for="user_active">Activate User</label>
```

想想标签功能的强大吧，它们可以让用户更轻松地跟表单互动。

标签可以将文字标签同表单域结合起来，同时你还可以为元素制定快捷键。如果你在表单中将一个标签指向了搜索框，那么当快捷键被按下的时候，鼠标指针就会被放置在搜索框的里面：

---

① 如果你将一个容器元素的位置设置为相对（`poistion:relative`），那么这个容器中的绝对定位的元素的原点就是该容器的左上角。

accessibility/index.html

```
<form id="search_form" method="get" action="/recipes/">
  <div>
    <label for="search_keywords" accesskey="s">Keywords</label>
    <input type="text" id="search_keywords" name="keywords">

    <input type="image" alt="Search" src="images/search.png">
  </div>
```

在这里用 S 作为快捷键，但是实际的使用组合可能不同，具体视用户的浏览器和操作系统而定。Mac 的 Safari 和 Firefox 需要你同时按下 Ctrl+S，而在 Windows 里，这一套快捷键对应的是 Save，所以你要在 Firefox 里用 Shift+Alt+S，在 IE 里用 Alt+S。

搜索表单的 `label` 标签带来了一些额外的文字，我们并不需要这些文字。所以我们会用 CSS 来隐藏它，就像刚刚对导航跳转链接做的事情一样：

accessibility/stylesheets/layout.css

```
#search_form label{
  position:absolute;
  left:-9999px;}
```

如果想要它变得更好，你可以尝试着将几条 CSS 规则并成一条。

到目前为止，我们对页面做了很多修改，这些修改极大地提高了可访问性。你可能还需要在导航条上添加一些跳转链接，进一步提高网页的可访问性。实际上，你肯定要为移动用户添加这些功能的。

小乔爱问……

等等，如果每个浏览器都有不同的键盘快捷键，那么用户是如何判断该用什么呢

屏幕阅读器（如 JAWS）会在遇到快捷键的时候向用户读出快捷键，这对使用屏幕阅读器的用户来说相当有用。然而你的快捷键并不会自动出现在大多数浏览器中。

你可能需要在页面上放一个“可用性声明”，然后通过跳转链接导航，通过这个链接链到可访问性声明和快捷键列表，同时还可以在这个页面上加上其他对残疾人用户导航有用的信息。同时这个页面上还需要有一个表单，它可以收集用户的反馈，从而让你对这些东西的使用情况有个大概了解。

16.6　使用制表键

现在 Foodbox 网站还没有特别复杂的表单，这只是时间问题。你总是会需要创建一个复杂表

单的。当你面临这种任务的时候，要想想用户可能使用什么样的快捷键来浏览表单。当用户按下制表键的时候，鼠标指针会从一个元素跳到另外一个元素。但是如果表单没有经过精心设计，鼠标指针可能在页面上乱走。可以使用 `tabindex` 属性进一步控制表单域。

`tabindex` 属性可以让你在界面上控制标签的顺序，只需将表单里的每个区域的 `tabindex` 属性依次递增就可以了。你可以为任何可交互元素设置 `tabindex`，像链接、下拉列表、文字输入框、多选框和单选框等都可以。

接下来，将 Foodbox 注册用的表单变成一个具有可访问性的表单：

```
<form id="signup" action="/signup" method="get">
  <p>
    <label for="account_login">Login</label><br>
    <input id="account_login"
           name="account[login]"
           size="30" type="text" tabindex="1">
  </p>

  <p>
    <label for="email">Email</label><br>
    <input id="account_email"
           name="account[email]"
           size="30" type="text" tabindex="2"></p>

  <p>
    <label for="password">Password</label><br>
    <input id="account_password"
           name="account[password]"
           size="30" type="password" tabindex="3">
  </p>

  <p>
    <label for="password_confirmation">
           Confirm Password
    </label><br>
    <input id="account_password_confirmation"
         name="account[password_confirmation]"
         size="30" type="password" tabindex="4">
  </p>

  <p>
     <label
       <input class="button" name="commit"
              type="submit" value="Sign up"
              tabindex="5">
     </label>
     or <a href="/" tabindex="6">Cancel</a>
  </p>

</form>
```

竖直符（和其他的特殊字符）

如果你在页面上用到了通道符号[①]（在首页的页脚中就用了），那么就可能会对屏幕阅读器处理这种字符的方式感到惊讶。Foodbox的页脚在一个屏幕阅读器眼中是这样的：

“Copyright copyright two thousand eight Foodbox, LLC, all rights reserved. Link Terms of Service vertical bar Link Privacy Policy.”

它将版权符号读成了一个词，所以“copyright”这个词被读了两遍。另外竖直符也被识别出来了。

这对一个小网站没什么，但是请想想，如果页脚里有6个被竖直符分开的信息，阅读器会读成什么样子。所以当你开发网站的时候要对这些特殊字符特别小心，尤其是那些跟内容没有直接联系的字符。你能想到一个更好地处理页脚的方法吗？

避免`tabindex`[②]

你可能觉得这种说法会引起口水战。不要担心，这种事情很平常。制定`tabindex`是一件令人崩溃的事情，特别是在一个复杂且有众多域的表单里，向表单中部添加内容的时候，`tabindex`带来的麻烦尤甚。如果你在设计表单的时候用心，完全可以避免使用`tabindex`。如果允许表单自上而下，`tabindex`自然没有问题，你完全不必操这份心。但是还要记住，在多个平台下的多种浏览器中用键盘测试表单。

16.7 可访问性清单

在发布网站之前，你需要做一个所有页面的快速验证工作。验证工作应该包括以下几条。

- 确定所有页面的标签都是合法的。
- 确认所有的样式表都是合法的。
- 看看是不是所有的图片标签都加上了有用的、描述性的默认文字。
- 在基于文本的浏览器（比如，Lynx）中检查所有页面是否显示正常。
- 检查页面在老式浏览器中是否显示正常。
- 关闭JavaScript，看看页面是否可用。
- 在较慢的网速下看看网页需要多长时间才能加载。

① 即“|”。——译者注
② 使用Tab键切换焦点时的顺序。

- 关掉图片显示功能，确保没有哪个文字是仅仅以图片的形式存在的。
- 在 Firefox 上安装 Fangs 这个扩展，看看屏幕阅读器是如何解析页面的。
- 想办法弄到 JAWS[①]或者 Window-Eyes[②]的试用版，将显示器关掉，然后尝试浏览你的网页。
- 找第三方检查页面内容的拼写、语法和其他相关问题。
- 找到所有的“单击这里”的文字，因为并不是所有的用户都有鼠标。另外，现在的用户已经知道面对超链接的时候该怎么做了。所以如果你的链接还不够明显，那么请检查你实现链接的方式。
- 用 Colorblind Web Page Filter[③]一类的软件来测试页面在不同色盲症患者眼中的样子。另外 Color Oracle[④]也是一款类似的优秀的跨平台软件。
- 确保页面上没有快速闪动的元素，这可能引起癫痫用户的不适。
- 确保添加了跳转导航，以帮助使用屏幕阅读器的用户可以跳过重复的导航。
- 确保用户可以在表单域间轻松地切换。
- 确保所有的视频都有说明文字或者字幕，以帮助听力障碍用户正常使用。

16.8 小结

可访问性和可用性十分重要，是不能忽视的。即便不面对残疾人用户，这些可以帮残疾人用户正常访问网站的技术也能帮助你让网站对更多其他用户可用。一定要记住，可访问性并不是*只针对盲人用户的*。实际上，可访问性指的是能让任何设备上的任何用户使用网站。无论处于开发的哪个阶段，你都应该将这条定义当作必须实现的目标。

① http://www.freedomscientific.com/。
② http://www.gwmicro.com/。
③ http://colorfilter.wickline.org/。
④ http://colororacle.cartography.ch/。

第 17 章

制作收藏夹图标

大多数浏览器都会在网页的地址栏旁边显示一个小图标，这个图片还会显示在收藏夹网址的书签旁边。我们把这个图标叫做收藏夹图标（favicon）。在支持多标签的浏览器中，收藏夹图标会出现在页面标签上。这种统一样式的曝光能加强网站的品牌建设。大多数受欢迎的网站都有收藏夹图标，我们的 Foodbox 也不例外。

17.1　创建简单的图标

一个成功的收藏夹图标应该可以反映我们的品牌。我们可以使用 Foodbox 的 Logo 中的四个方形。打开 Photoshop，新建一个文档，文档大小为 64 像素宽，64 像素高，分辨率是 72dpi，色彩空间则设为 RGB。背景颜色是白色。

单击 File 菜单里的 Place 选项，将 foodbox.ai 文件导入。然后用缩放手柄调整图片大小，直到四个方形刚刚好是画布的大小（参见图 17-1）。在调整大小的时候记得按住 Shift 键不放，不然可能会导致 Logo 的比例变形。

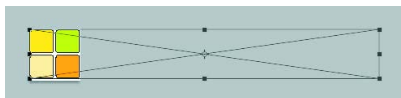


图 17-1　调整 Foodbox 的 Logo 大小，让四个方形都在画布内部

17.2　创建收藏夹图标

网页收藏夹是一个 16 像素 × 16 像素的 windows.ico 文件，Photoshop 并没有内建可以导出这个格式文件的滤镜。但是你可以从 Telegraphics 弄到一个免费的滤镜，并且这个滤镜支持几乎所有版本的 Photoshop。[1]另外还有一个命令行工具叫做 png2ico[2]，该工具可以将 PNG 文件转换成

① http://www.telegraphics.com.au/sw/。

② http://www.winterdrache.de/freeware/png2ico/index.html。

ICO 文件。Windows 用户可以下载它的二进制安装文件，Linux 和 Mac 用户则需要看看它们各自的包管理器是如何安装这个软件的。

在 Photoshop 中用 Image Size（图像大小）命令将图形改成 16 像素×16 像素大小，该命令在 Image（图像）菜单下。记得勾上 Bicubic Sharper[二次立方（较锐利）]选项，这个选项让图片在缩小的时候保持清晰。

将文件保存为 favicon.png，因为收藏夹图标支持透明图片，所以保存成 PNG 格式是没有问题的。

打开终端或者是命令行提示符，进入包含这个文件的文件夹，键入以下命令：

```
png2ico favicon.ico favicon.png
```

然后将 favicon.ico 文件保存在网站的根目录下（注意，不是 images 文件夹），这样一来大多数浏览器都能自己找到这个文件。在本地系统里你是看不到这个收藏夹图标的，所以必须将网站传到服务器上，另外可能还要重启服务器，才能让它显示在地址栏旁。在图 17-2 中，你可以看到最终的效果。

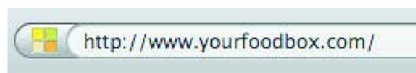


图 17-2　我们的收藏夹图标

Safari、Firefox 和 IE 7 都支持收藏夹图标，只是在 IE6 上你可能需要一些额外的功夫才能让它正常显示。

如果图标不能正常显示，请将以下的代码加入首页的 head 区域中：

```
<link rel="shortcut icon" href="favicon.ico">
<link rel="icon" type="image/ico" href="favicon.ico">
```

17.3　小结

收藏夹图标是网站品牌的一个重要部分，它能够让用户牢记住你的品牌，因为在他们访问网站时能看到这个图标。即便是他们在访问其他页面的时候，也总是能够收藏栏中的图标。选用网站的 Logo 等有代表性的图片作为收藏夹图标，一定会事半功倍。

第 18 章

搜索引擎优化

Foodbox 网站已经就位了，所有人都觉得它看起来不错。没有什么能比全力完成一个项目更能让你高兴了。但是，建起网站只是一个开始，如果网站没有人来看，那么它会变得毫无意义。因此我们需要在这里谈谈如何能让网站更容易被搜索引擎找到，同时又不损害它的用户友好度。

18.1 内容为王

关于这个概念我已经说过多次了，但是我仍然要再强调一遍：无论网站多么漂亮，如果没有好内容，它对用户不会有多少吸引力。以 Flickr[①]为例，这个网站有着保守且平淡无奇的界面，设计最简化，用尽可能少的颜色，因为开发人员知道用户到这个网站来是为了看相片的，相片就是 Flickr 的内容。

在搜索引擎看来，内容也是最重要的。引擎希望网站的内容跟网站的关键字是相匹配的。

18.1.1 “欺骗”搜索引擎

首先我要发表一个免责声明：如果你在看这节的时候认为这些技巧可以帮你提高网页排名，那你就大错特错了。搜索引擎其实对这些小把戏心知肚明。我在这里列出这些小把戏，目的是让你能够认清它们的面目并远离它们。这其中有些东西可能是你无意识加上去的，而另外一些则可能是那些无良的 SEO 公司的把戏，或者为了满足客户要求而不得不做的事情。这些技巧有时候会在短期内起一些作用，但长远来看，它们一定会对你的网页造成严重伤害。

1. 关键字重载

所谓关键字重载，基本上指的是往你的页面上塞入无穷尽的关键字。有些人这样做是因为他对自己输入的关键字并没有概念，还有些人输入大量重复的关键字是想引起搜索引擎的注意。大

① http://www.flickr.com/。

体上说，每个关键字不应该被重复两次以上，并且每一页的关键字应该保持在 30～45 个的范围。

2．不相关的关键字

如果你用了那些跟页面内容毫不相关的关键字，那就有大麻烦了。有些网站专门用报纸头条标题里的词组，或是用 Google 搜索排行里头几名的词做关键字，他们觉得这样做可以让更多的用户访问他们的网页。

适合 Foodbox 的关键字有：鸡肉、火鸡、菜谱、晚饭和意大利面。不太搭边的关键字则类似免费 MP3、视频、免费 iPod 或者是帕丽斯·希尔顿[①]。

别笑，这种事情经常发生。特别是很多老板会要求程序员这样做——当发生这种事情的时候，你需要想清楚，这到底是不是你愿意做的事。我是不会愿意的，因为我不想在上线几个月后 Google 把网站列入黑名单的时候受到牵连。

3．内容置换

这个技巧通常依赖于服务端的技术——当探查到搜索引擎访问的时候，服务端会提供一套另外的内容，跟用户访问时看到的东西完全不同。通常，搜索引擎会用爬虫程序抓取页面的内容，用关键字、内容和链接建立索引，然后存到自己的数据库。用另一套内容来糊弄它们是一种不诚实的做法，而最终搜索引擎会通过用户的投诉知晓这类不诚实的举动。毕竟搜索引擎也是一项商业服务，它们看重的是搜索结果要跟搜索条件有一定的相关性。

4．内容隐藏

这是一种老旧的手段，但是它的受欢迎程度还是让我感到惊讶，而且新手特别喜欢使用这种方式。此方法会将大量的短语和关键字放在内容页里，然后用某种技巧将它们隐藏起来。以前人们把这批文字设置成跟背景同一颜色，后来搜索引擎能够识别这个把戏了，他们又用 CSS 将内容放到页面之外（即“负位置”方法）或是其他方式来隐藏这些内容。

18.1.2 到底什么是内容

内容指的是用户来到页面时想看到的东西。文字显然是内容的一种，另外还有图片、视频、音乐和可下载文件。通常来讲浏览器对这些东西都感兴趣，所以你要做的就是帮助浏览器去找到这些内容。

你已经学会了用 `alt` 标签给图片加上默认文字，而你可能不知道的是，搜索引擎因为无法分析图片，所以它们也会依赖描述图像的默认文字，就跟屏幕阅读器一个样。所以要保证默认文字

① 美国著名模特、歌手、演员。——编者注

跟图片内容一致，且具有描述性。

18.2 选择关键字

毫无疑问，Foodbox 需要一些关键字，但是想要弄清楚用户到底通过什么找到这个网站，就比较困难了。这节里我会提供一些方法让你和你的团队建立一个关键字列表。

18.2.1 猜想他们是如何找到你的

你认为用户搜索你的网站的时候，会用哪些词？记下来。比如说当我想到这类网站的时候，我认为用户会搜索食物、烘烤、下厨、菜谱、简单的晚餐、零食、甜点和做菜指导。

18.2.2 决定你想如何被发现

再写下你觉得网站应该出现在那些词的搜索结果中。上一节提到的几个还不错，但是你总能想出更多来。一个本地客户可能会希望你加上城市、州（省）名或者区域名，这样网页用户可以在搜索“菜谱，北京”的时候找到你的网站。

小乔爱问……

Flash 怎么办

Google 和 Adobe 公司达成过合作协议，这使得对 Flash 影片的索引成为可能。[1]但是索引效果的好坏取决于 Flash 影片的作者是如何创建 Flash 影片的。比如，如果 Flash 里含有文字和链接，那 Google 有可能会看到它们。另外，Flash 也是被包含在 HTML 页面里的，所以你可以用 `meta` 标签来为它加上关键字和简短的描述。

在将来，可能会有更多的搜索引擎具备搜索 Flash 的能力——毕竟它在可访问性方面也已经作了很多努力。时代总是在不断变化，作为开发者，你需要与时俱进。

18.2.3 看看竞争对手

看看你的商业伙伴和竞争对手，他们都用了什么样的关键字。查看他们页面的源码就跟查看自己的一样简单，所以还是去看看他们的用法吧。但是千万不要照抄他们的东西，也不要将他们用过的短语重组使用。曾经碰到一个客户，硬要我把竞争对手的名字加到自己的关键字里去，这

① http://searchengineland.com/google-now-crawling-and-indexing-flash-content-14299。

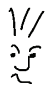
样做是不道德的。随着你做的网络开发越来越多，特别是你对设计和内容两个方面得心应手之后，你跟客户打交道的能力也会越来越强，也就会更完美地处理这种奇怪甚至是危险的要求了。

18.2.4 添加关键字

将 index.html 文件打开，在 head 区域内加上 meta 标签：

```
<meta name="keywords" content="foodbox, recipes, cookbook, desserts,
entrees, dinner, share, browse, ingredients, mexican, italian, community">
```

name 属性决定了 meta 标签的类型，而 content 属性则指定了内容和值。输入关键字的时候要用逗号把词语或者短语隔开。



小乔爱问……

关键字有那么重要吗？我的一些对手一个关键字都没有，可他们的页面却表现得很好

如果使用正确，关键字还是很有用的。你需要做的是保持关键字和页面内容的一致性。很多搜索引擎会因为关键字同内容不符而大大降低一个页面的权重，因为这是一种欺骗行为。

你对手的网页没有关键字，却在搜索结果里排名很高，这里面的具体原因可以有很多。CodingHorror[①]这个网站一个关键字都没有，但是它会保持内容的更新，而且还有很多回链。对页面的链接所带来的权重在搜索引擎的等级系统里要大大超过内容。

但你并不是 CodingHorror，无法依靠大量的流量和回链。因此，如果希望优化页面，那么不要在关键字上犯错误。只要它们跟内容是相符的，网站的等级就不会受到危害。

18.3 完善页面内容

现在关键字选好了，你要做的是调整页面内容，要关键字零星地出现在页面中。如果你有一个不错的网站编辑，那现在正是他大展拳脚的时机；但大多数情况下，是你必须自己做这件事情。在页面内容里找到几个空档，然后把关键字插进去。完成之后，尝试用以下技巧校对。

- 从后往前读你的文字。这可以帮你找到拼写和标点错误，因为这个时候读出来的词是没有语境的。
- 大声朗读文字。如果你没有发笑，就说明文章还不错。你还应该在几个朋友面前读一读，收集反馈。

① http://www.codinghorror.com/。

为了巩固网页在搜索引擎中的地位，不仅要保持内容的一致性，还要保证关键字和页面元素的一致性。

每个页面都应该有一个自己的 `title` 标签——标签内容由该页面的标题和网站名字组成。这样一来，网页最重要的信息（标题）就会出现在搜索引擎的搜索结果里面。几乎所有的搜索引擎都会在搜索结果中显示页面标题。

另外，每个页面至少都应该有一个 `h1` 标签，标签里应该是页面标题。这种一致性将有助于搜索引擎来判断页面内容的强度。

最后，要从标题里抽取一些关键字。从页面的链接或者是从段落里取一些关键字也是可以的。

18.4　不要因为优化而忽略了用户

这些内容上的调整可能会引来令人不快的后果——流失用户。要记住，内容是最重要的，所以不要因为一点点关键字就牺牲好内容。内容首先是写给用户看的，其次才是给爬虫看的。

18.5　你和链接

页面上链接数量的不同会给搜索引擎排名带来的印象不同，有好有坏。如果页面上有太多指向外部站点的链接，搜索引擎可能会认为页面上的内容没有关联性，从而降低对页面的评分。

另外，也要注意其他人是用什么方式链接到你的页面的。通常情况下，你是会期待别人外链到你的网站上的，这样一来搜索引擎会认为由于你的页面上内容关联性很好，所以别的网站才会给你输送流量。有些网站，像 link farms、link exchanges 之类的，Google 会认为它们是*不太好的邻居*——如果在这些网站上有你网站的链接，搜索引擎也会降低页面的评分，因为你可能会被认为是垃圾信息发送商。

当然你无法控制别人如何链接到你的网站，但是有时会有些你不知底细的人来要求跟你交换链接，那么在跟人交换链接的时候，要考虑清楚这个交换是不是给你们双方都带来一样的好处。

18.6　到最后其实都是常识

如果你读完了这一章并且觉得“这章并没有讲什么新东西啊”，那你算是上道了。关于 SEO，并不存在什么魔法。想要一下子爬到 Google 搜索结果的顶端且保持很长一段时间，是不太可能的。有些人可能会蒙蔽系统一阵子，但是本章谈论的话题是*搜索引擎优化*，而不是*搜索引擎欺诈*。

如果页面内容写得不错且可以保持更新，对用户有帮助，并且为你赢来了一些外链，这些外

链将会帮助你提高网站在搜索引擎中的排名。所以，下次你的老板或者客户想要雇佣一个“SEO专家”的时候，建议他们雇佣一个优秀的网站编辑，负责编写页面内容。如果你开发的网站确实有一些东西吸引用户，并且愿意做外链，那所谓的优化的结果就会自己找上门来。

18.7　小结

本章中你学会了如何在不影响可用性的前提下提升页面对搜索引擎的可见度。搜索引擎优化对你来说是一个长久的任务，因为规则时时在变。

第 19 章

针对移动设备的设计

Foodbox 的老板又在召唤你了。他们发现不管到哪儿，都有人在用黑莓、iPhone、Windows Mobile 或者其他移动设备上网，所以他们希望你能提供一个移动版的 Foodbox，好让用户在杂货店里买东西的时候也能通过网站来查菜谱。

只有了解移动设备的用户，才能做好适合移动设备的设计。如果你真地确定要照顾到这一类用户，那你先要去了解一些不同的移动平台，然后再确定哪个平台是最适合你去支持的。这个过程有助于制定一个可以实施的移动平台计划。

19.1 移动用户

移动用户同桌面用户和笔记本用户的需求不同。

首先，移动用户浏览网页的目的不一样。他们不会用手机上过于小巧的按键在页面上输入整个菜谱，但是却有可能在杂货铺搜索今晚晚餐的食材。所以你可能只需要为移动用户提供一个现有网站功能的子集就够了。

其次，移动网络的网速总是很慢，而用户会期待页面载入速度不要太慢。在我写这本书的时候，3G 服务在美国还没有完全普及，大多数的数据套餐都很慢很慢。

再次，用户希望在移动设备上浏览网页的时候，网页可以适应小屏幕的阅读环境。在 iPhone 里，Foodbox 的设计被完好地保留，但是如果不放大，用户很难看清页面上的内容，如图 19-1 所示。另外，Opera 移动版也保留了页面的样式，但是你几乎看不清任何文字（参见图 19-2）。

最后，应该让用户能够轻易地发现他们正在查找的东西。对于使用手机键盘的用户来说，专为普通用户设计的那一套导航结构显得过于笨重。

在移动版的 Foodbox 里，我们将为用户解决这几个问题。



图 19-1 iPhone Safari 里 Foodbox 的模样



图 19-2 Opera 移动版中的 Foodbox

19.2 关于（很）小屏幕

如果人人都用 iPhone，那开发人员不用费太多工夫就能让页面正常运转。iPhone 内置了一个强大的浏览器，有着不俗的页面呈现能力和对 JavaScript 的支持。但是大部分人都不用 iPhone，大多数人用的都是有着 2～3 英寸[1]屏幕的手机，这些手机往往有一个不支持脚本语言的专用浏览器。小尺寸的屏幕让你没有多少空间可以用来显示信息，这意味我们不得不抛弃一部分现有页面的内容。

比如，在一个正常的浏览器中，你可能会把每一页的导航都重新设计一下。但是现在不得不抛弃这些东西，因为没有足够的屏幕空间。当然，这并不是说用户不会在页面到处看——我们需要做的是将导航菜单换成简单的不带样式的页面列表。保持页面的朴素可以减少下载和加载时间，并且可以为页面其他内容节省出宝贵的空间。

说到小尺寸屏幕上的空间节省，不要忘记那些跟品牌有关的元素。首先你一定要换一个小一点的页头图片，要么把它换成文字，要么用你学到的缩放技巧缩小 Banner 的体积（参见第 17 章）。

[1] 1 英寸=2.54 厘米。——编者注

为了在小屏幕上显示出更多内容，一些设备会将导航隐藏。所以你可能需要设计自己的后退按钮，以简化移动版页面上用户的浏览过程。

最后，要记住保持文字的可读性。大多数用户在看不清内容的时候，不会眯着眼睛读，而是直接跳过去，并且多数设备是没有缩放功能的。

19.3　JavaScript

很多移动设备不支持 JavaScript。虽然 iPhone 支持 JavaScript 并且很依赖它，但是现在市面上大部分黑莓手机完全不支持 JavaScript。我其实并不太担心这个问题。因为我从来没有提倡过将页面的重要功能单独用 JavaScript 实现，除非你能提供一种备选方案——你应该还记得屏幕阅读器对 JavaScript 也不是很友好。

19.4　提供移动版

有很多方法可以让你为用户提供适合移动设备的内容。比如你可以为移动设备单独设计一套样式表，然后通过判断用户代理（user agent）来决定提供哪些不同的样式表给不同的用户。你还可以将移动版的内容放在一个单独的 URL 里面。

19.4.1　移动版样式表

CSS 标准为媒体类型（media type）属性提供了一个值叫做 `handheld`，这个值是为了样式表在移动设备上的使用而设计的。在 14.2 节中，为打印功能专门设计了一套样式表。现在你可以用同样的方法来为移动设备准备一套样式表。乍一看，这是一个最好最直接的方式，但是它有一个致命的问题：没有人用这个属性值。

iPhone、iPod Touch 和 Windows Mobile 试图为用户提供“真实”的互联网体验，所以它们会去加载那些为桌面客户端准备的样式表。

19.4.2　用户代理探测

很多用户将用户代理探测和后台技术（或服务器配置）结合起来，为不同平台上的用户提供不同的设计。这似乎是个不错的主意，只是有些用户可能不会这么认为。

比如，Apple 的开发者指南就不建议 iPhone 开发者为 iPhone 提供一套全新的页面。因为如果用户看到的不是他们期待的网站而是一个新页面，他们会觉得奇怪。Apple 建议的是，当检测到用户在使用 iPhone 的时候，给他们一个连接到优化页面的链接，然后让用户自己决定是要用优化过的页面还是原始页面。Twitter 和 Amazon 都为用户提供了回到原始页面的链接。当用户通过

移动设备访问 Foodbox 的时候，你应该为它建立一个 cookie，等用户下次再通过移动设备访问网站的时候，帮他们自动跳转到移动页面。

19.4.3　使用子域名

你也可以在子域名下架设移动版网页。这个方法要求用户知道移动版网站的地址，也就是说，你需要在主页上为这个地址大做广告。

为了不产生额外的文件，可以将移动版的地址指向原版页面的文件，然后通过服务器端的代码做判断，如果是从子域名过来的链接，则提供相应的样式表，为移动用户呈现一个完全不同的版式。我更倾向于用这个方法。

19.5　做决定——到底要支持什么平台

市面上的移动浏览器有很多，它们显示页面的方式各不相同。你必须做出决定，看看将主要精力放在哪几个上面，还有哪些是不用太操心的。我建议你把之前挑选桌面浏览器的流程再走一遍，记住要选那些最受欢迎的。另外，还要搞清楚你的客户或者老板用的什么型号的手机，一定也要提供对这些手机的支持！

测试移动设备上的设计

你可能有真实的设备去测试网站，也可以用那些很受欢迎的平台的模拟器来测：

- BlackBerry
  http://www.blackberry.com/developers/downloads/simulators/index.shtml[①]
- Google Android
  http://code.google.com/android/reference/emulator.html
- iPhone
  http://marketcircle.com/iphoney/[②]
- Opera Mini
  http://www.opera.com/mini/demo/
- Windows Mobile
  http://www.nsbasic.com/ce/info/technotes/TN23.htm

① 黑莓提供了很多款手机的模拟器，但是网站上并没有说哪个模拟器对应哪个手机。

② 在写这本书的时候，iPhone 只能在 Mac 上运行。安装了 Safari 的 Windows 可以在 http://www.testiphone.com/上找到不错的模拟器，但是没有用户代理探测这个功能。

AdMob 网站提供了一些非常好的数据[1]，可以根据这些数据来判断哪些移动平台是最受欢迎的。AdMob 提供服务给开发者，让它们把广告投放到移动软件和网站上，它并不是唯一一家，却是最大的一家。通常情况下，如果你还没有开始收集自己的数据，那么还是需要依赖一个第三方数据源作为参考的。

AdMob 的数据资料显示大部分用户使用 iPhone，但我们还是做一个可以在几种不同设备上运行的页面吧。

现在我们了解到，大多数手机不支持 JavaScript，而且屏幕也很小，用户期待可以快速地下载内容。

19.5.1　在不产生重复内容的情况下制作一个镜像

最有效的流失用户的方法就是建立一个移动版本的网站，但是不为它更新。这本书写到现在，都是在用 HTML 实现我们的设计，并没有涉及任何服务器端的技术。但是，我一直假设的是这本书的读者是一个开发人员，你应该具备一些建设动态网站的知识。

在这个前提下，为网站设立一个新域名，并把它指向主域名。然后再用一些服务器端的逻辑来判断用户请求的 URL。本书中，我会教你如何用 PHP 将静态的网站转换成一个适合移动设备的页面。然后，你根据自己网站的不同设计，可以按照这个过程修改自己的域名。

这本书并不会真正涉及运维方面的东西，因此你的主机提供商仍然负责帮你设置多个域名。你应该在 DNS 上建立一个新的记录，将子域名指向主域名的 IP：

```
www.yourfoodbox.com  A  12.34.56.78
m.yourfoodbox.com    A  12.34.56.78
```

一些主机服务商（像 Dreamhost[2]）会提供很简便易行的方式设置域名的镜像。你可以找服务商、系统管理员询问如何设置域名，也可以看服务器的说明书。无论怎样，你要做的都是将两个域名指向同一台服务器。

如果是在本机测试，那么可以在本地的 hosts 文件中做出如下改动：

```
127.0.0.1 localhost www.yourfoodbox.dev   m.yourfoodbox.dev
```

如果用的是 Windows 机器，hosts 文件在 c:\windows\system32\drivers\etc\hosts；在大多数的 Linux 机器和 OS X 系统里，这个文件在/etc/hosts。修改这个文件需要系统管理员权限。

---

① http://www.testiphone.com/。

② http://www.dreamhost.com/。

19.5.2 调整内容

有好几种方法可以调整页面内容使之适合移动设备，但是仔细研究一下，会发现这些方法都有问题。不能用 handheld 属性值来制定样式表，因为大多数浏览器都不承认这个值，也不会去找为大屏幕准备的样式表，因此这个方法行不通。我们也曾讨论过移动用户会更加专注于内容，同时他们的连接速度不是很快。这就需要将发送出去的数据的体积尽量缩小，这样页面的载入速度才会变快。我们需要抛弃无用的负担，为用户展示一个基本的页面。

在关掉样式的情况下，页面在手机上的可用性出乎意料地好。我们还可以禁止所有的图片，并将图片形式的超链接换成普通的文字连接。于是我想到早在 2005 年，一个叫做麦克・戴维森所创造的一个极具创意的想法可能会是最适合我们的方法。[①]

麦克的方式是这样的，它用 PHP 处理所有的 HTML 页面，挑出里面的内容——但是这种情况只有在为移动页面准备的域名下才会发生。只用做一点小小的修改，他的这个方法就能迅速用在我们的页面上，为我们创造一个移动版本的页面。为了使用这个方法，你需要一个 Apache 软件，并且将它配置一下，使之能够解析 PHP 页。我假设你已经搭建好了这个环境。[②]

你要确保 Apache 是以 PHP 模式运行的，这样这个方法才行得通。同时，还要确认 PHP 是开启状态。如果主机上不是用 PHP 来运行 CGI 程序的，那这个方法也不会成功。如果想要确定一下自己的配置有没有问题，你可以在主机上新建一个叫做 info.php 的文件，并在里面加入这样的内容：

mobile/info.php

```
<?php phpinfo(); ?>
```

接下来，你会看到如图 19-3 上显示的内容，里面的 Servr API 应该是 Apache2.0 模块。[③]

这里需要一个特殊的目录，并告诉 Apache 所有对 HTML 的处理都要转移到 PHP 解释器中，这样过滤程序才能正常工作。改动网站根目录的.htaccess 文件可以实现这个要求，在文件里加入下面一行代码：

mobile/.htaccess

```
AddType application/x-httpd-php .html .htm
```

这样一来，PHP 解释器将会处理所有的 HTML 文件。

---

① http://www.mikeindustries.com/blog/archive/2005/07/make-your-site-mobile-friendly。

② Dreamhost（http://www.dreamhost.com/）提供便宜的主机计划，它们的主机已经包含了我上面说的这些环境。如果不想自己搭建，你可以去用它们的服务。我甚至申请了一个折扣码 WDFD，用这个码你可以拿到折扣。

③ 如果你对 PHP 一点都不了解，那就去学学吧。这并不是我最爱的语言，但是我认为每一个做网络开发的程序员都应该对它有所了解。这个语言很灵活、强大而且有广泛的支持。

PHP Version 5.2.8	php
System	Darwin coalcar.local 9.8.0 Darwin Kernel Version 9.8.0: Wed Jul 15 16:55:01 PDT 2009; root:xnu-1228.15.4~1/RELEASE_I386 i386
Build Date	Feb 5 2009 21:17:28
Configure Command	'/SourceCache/apache_mod_php/apache_mod_php-44.2/php/configure' '--prefix=/usr' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--disable-dependency-tracking' '--with-apxs2=/usr/sbin/apxs' '--with-ldap=/usr' '--with-kerberos=/usr' '--enable-cli' '--with-zlib-dir=/usr' '--enable-trans-sid' '--with-xml' '--enable-exif' '--enable-ftp' '--enable-mbstring' '--enable-mbregex' '--enable-dbx' '--enable-sockets' '--with-iodbc=/usr' '--with-curl=/usr' '--with-config-file-path=/etc' '--sysconfdir=/private/etc' '--with-mysql-sock=/var/mysql' '--with-mysqli=/usr/bin/mysql_config' '--with-mysql=/usr' '--with-openssl' '--with-xmlrpc' '--with-xsl=/usr' '--without-pear'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc

图 19-3　用 Apache 设置成功的 PHP

19.5.3　处理程序

这一节会用到异常强大的 PHP 特性，包括 autoprepend 和 autoappend。这个特性可以让你在每次页面请求之前和请求之后执行一些代码。可以用这个技巧来写系统日志，或者是设置一些保留变量。另外还可以用这个技术来捕获缓存下来的服务器响应，并对它加以解析。

这里以麦克的初始程序为蓝本，先写 prepend 的脚本。

新建一个叫做 global_prepend.php 的文件，在其中添加下面的代码：

mobile/global_prepend.php

```
<?php
function callback($b) {

  $mobile_domain = "m.yourfoodbox.com";
  $web_domain = "www.yourfoodbox.com";

  if ($_SERVER['SERVER_NAME'] == $mobile_domain) {

  // replace www.yourfoodbox.com with m.yourfoodb.com
  $b = str_replace($web_domain, $mobile_domain, $b);

  // replace all hyperlinked images with regular links, using the alt text
  $b = preg_replace('/(<a[^>]*>)(<img[^>]+alt=")([^"]*)("[^>]*>)(<\/a>)/i',
   '<p class="link">$1$3$5</p>', $b);

  // replace images with paragraph tags
```

```
    $b = preg_replace('/(<img[^>]+alt=")([^"]*)("[^>]*>)/',
     '<p class="image">[$2]</p>', $b);

    // strip out stylesheet calls
    $b = preg_replace('/(<link[^>]+rel="[^"]*stylesheet"[^>]*>|style="[^"]*")/i',
     '', $b);

    //remove scripts
    $b = preg_replace('/<script[^>]*>.*?<\/script>/i', '', $b);

    // remove style tags and comments
    $b = preg_replace('/<style[^>]*>.*?<\/style>|<!--.*?-->/i', '', $b);

    // add robots nofollow directive to keep the search engines out!
    $b = preg_replace('/<\/head>/i',
      '<meta name="robots" content="noindex, nofollow"></head>', $b);

    }
    return $b;

  }
  ob_start("callback");
  ?>
```

第 2 行的代码定义了一个把 HTML 页变成字符串的函数，接下来的两行定义了两个变量，里面存的是正常域名和移动版域名。第 7 行的代码将用户请求的域名同移动版域名做对比，如果一样，会把页面内容过滤一遍。

在第 10 行里，将页面上所有的正常域名都换成了移动版域名。这一步意味着你手写在网页里的域名也会被替换，以防止用户误入普通版域名。然后是第 12 行，这一行替换掉了所有带链接的图片，比如注册和登录按钮，具体替换方法是用原有图片里 `alt` 属性的值代替图片成为普通链接。第 16 行代码做的事情是将普通图片替换成用方括号括起来的 `alt` 文字。

从第 20 行开始，逐步去掉页面里的样式表链接、脚本调用、直接插入的样式和注释。这样做能有效减少向移动设备发送的数据量，从而提高页面的表现。有些用户在移动设备上收发数据是按照流量计费的，这个做法会让他们很开心。

第 30 行是最后一步，它在 `head` 标签结束之前加了一行声明，拒绝搜索引擎索引页面内容和页面链接。我们可不希望因为有同一个网页的两个版本而被搜索引擎惩罚。

页面内容过滤差不多完成了，最后函数返回了这个字符串。在第 38 行调用了 `ob_start` 方法，开启了输出缓存，并将 `callback` 函数作为参数传给了它，这样在请求结束的时候这个函数就会被执行。然后需要一个 global_pappend.php 文件来做内容输出，把这个输出添加在返回请求的

后面：

mobile/global_append.php

```
<?php
  ob_end_flush();
?>
```

这个文件将转换好的内容发送出去，并将之呈现成页面。如果没有这一步，就什么东西也看不到了。现在，只需要激活刚刚我们写的过滤器就好了，方法是在.htaccess 文件里加上下面两行：

mobile/.htaccess

```
php_value auto_prepend_file /home/yourfoodbox/yourfoodbox.com/global_prepend.php
php_value auto_append_file /home/yourfoodbox/yourfoodbox.com/global_append.php
```

其中的路径必须是脚本文件的绝对路径。往服务器上传完这些文件后，你就已经为移动流量作好了准备！

19.5.4 进一步改进

页面的移动版看起来还不错，但是仍然可以对它的可用性做一些改进。现在，虽然页面顶部有个跳转链接可以让用户直接看到内容，但是如果把登录和注册链接也放在这个区域，会员用户用起来应该会更方便。另外，这个做法对使用屏幕阅读器的用户也是个好消息。

FastCGI 中的 PHP

即便服务器用的不是 Apache 的 PHP 服务，你照样可以使用上面提到的技巧。但是不能再用.htaccess 来定义 auto_append 的值了，你应该把相关的目录写在服务器的 php.ini 文件中。

```
auto_prepend_file =
    /home/yourfoodbox/yourfoodbox.com/global_prepend.php
auto_append_file =
    /home/yourfoodbox/yourfoodbox.com/global_append.php
```

最后，在.htaccess 文件中设定让 HTML 页面由 FastCGI 处理：

```
AddType php5-cgi .html .htm
```

具体的 AddType 设置是跟 Web 服务器相关的，所以如果遇到问题，请参照相关文档，或询问系统管理员或是空间提供商。

19.6 为移动用户做重构

这章为你提供了一个快速的解决方案，但是对于一个复杂的网站来说，这种程度的解决方案

是远远不够的。我们刚刚讨论过移动版用户的独特需求，以及对于他们来讲哪些内容是不需要的。我们可能需要借用服务器端技术，根据不同的用户代理返回不同的界面。这些内容超过了本书的讨论范围，但也不是那么困难。特别是如果网站有一个内容管理系统或是背后有一个服务器支持的话，从服务器端来解决移动版的问题会更加容易。

19.7 小结

移动用户正在快速增长，并且它们拥有完全不同于桌面用户的需求。随着类似 Google 和苹果手机技术的普及，将可以为用户提供更丰富的体验、更多样的互动。现在，你已经具备了相应的知识和使用相关工具的技能，可以在几乎所有的手机平台上，为大部分用户提供一个可用的移动网站。

第 20 章 测试与性能优化

再好的设计也会被很糟糕的实现打败。我们花了大力气来打造一个稳固的网站，如果用户需要很长时间才能载入我们的页面，那将是个很大的遗憾。如果能确定可能发生问题的地方，那么我们就有能力提高网站的性能。

20.1 优化性能的策略

性能优化指的是在用户那里页面能够更快地载入。在服务器端，可能会用到像缓存动态构建页面的方法。当将服务器的性能压榨完之后，仍然可以从页面设计方面入手，找到一个可能在客户端影响性能的因素。幸运的是，对这些问题的定位和解决过程是相当简单的。

首先，看看 HTML、CSS 和 JavaScript 文件的大小。这些文件里所包含的字符数的多少将直接影响到传送这些文件到客户端所耗费的时间。你需要将它们的体积缩减到最小。你也许会觉得这有点微不足道，但是如果服务器一天有数千请求，那这些字节的累积量将会相当可观。

然后，检查一下页面上的图片大小。在早些时候我们谈到过图片优化，但是在把图片正式放到网页里的时候，我们会经常忘记压缩和缩放图片。另外有些图片类型的压缩量，可以远超 Photoshop 所能提供的最大值。

另外还要检查浏览器的文件请求量，如果页面上链接有 3 个样式表、2 个 JavaScript 脚本文件和 5 张图片，那么一次访问将会给服务器带来 11 次请求。用户的客户端首先会把 HTML 文件下载下来，然后再向服务器发出别的所需文件的请求。

从另一个角度来看，用户可能不会发出那么多请求，因为大多数的图片和脚本文件都会缓存在他们的本地机器上。但是，如果页面上的东西过多，这些用户也有可能碰到载入速度过慢的情况，因为还有个特殊情况需要考虑，那就是 HTTP1.1 标准[①]。这个标准推荐在同一时间点，浏览

① http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.4。

器只能对服务器发出两次请求。

最后，要挑出那些不会经常改动的文件，以便采用某种措施来防止用户的浏览器过于频繁地检查这些文件是否有更新。比如说，如果你刚刚完成新一版的设计，那么大可放心地告诉客户端，把 Logo、样式表和脚本语言的缓存时间加长吧！

现在回到 Foodbox，看看如何检查上述策略。

20.2 确定影响性能的因素

不提倡过早开始做性能优化方面的工作，但同时也强调要对可能出现的性能问题有所了解，不然开发过程也会出毛病。有两种方法可以让你以第三者的角度来检查页面上的潜在问题。

注意，使用这两种方法的前提是已将这些页面上传到服务器，可以通过 Internet 访问到。

20.2.1 速度测试

WebSiteOptimization.com 提供了一套服务，它可以扫描任何一个 URL，然后确定一个页面（以及页面上所有元素）的大小和下载次数。这套服务算得上是一个好的开始。在 Firefox 中 Web Developer 插件里的 Tools 菜单下有个“Speed Test”（速度测试）的选项，你可以用它来对页面进行速度方面的检查。

检查结果显示网站上有 59 KB 的图片。作为对比，可以看到在微软的主页上有大概 61 KB 的图片，Adobe 的主页上约有 156 KB 的图片。在这一项上我们领先于他们。

这个测试还说 CSS 和 JavaScript 文件仍有压缩空间，并提醒我们有些图片没有设定长宽。长宽问题是在 HTML 文档中设定的，虽然这看起来并不是一个大问题（因为我们把多数图片放在了样式表里），但是修复这个问题可以让页面的载入速度看起来变得快一些，因为这样浏览器就可以无需依赖样式表而独自确定图片的长宽了。另外，在 HTML 文档中定义图片的宽和高可以让浏览器同时载入图片和文字。所以，要为 `img` 标签加上 `height` 和 `width` 属性以确定图片的宽和高。

测试结果中真正让人吃惊的是拨号用户的页面访问时间，根据测试报告，一个 56Kbps 的拨号用户需要 15.59 秒来完全载入主页上的图片、样式和内容。这个时间听起来很长，但是比起微软的主页仍然是小巫见大巫——它需要拨号链接的用户等上 56 秒还多！[①]

① 不要天真地以为这种用户不存在，有很多贫穷的地方都没有高速互联网。

20.2.2 YSlow

速度测试带给我们很多信息，但是并没有在解决问题方面给出太多的建议。倒是雅虎推出的一个 Firebug 插件（是的，这是一个插件的插件）可以为页面打分，然后告诉我们导致页面速度缓慢的原因。这个插件叫 YSlow。

YSlow 可以给页面打分，并且会提出一些性能方面的改进意见。或许你无法解决 YSlow 提出的所有问题，但是如果可以解决掉那些常见问题，效果也会非常好。

YSlow 生成的第一份报告说我们可以尝试一下几种技巧：用 ETags，压缩并缩小脚本文件，还可以看一看缓存过期时间的设定（参见图 20-1），在接下来的几节中就会讨论这几个方面的问题。

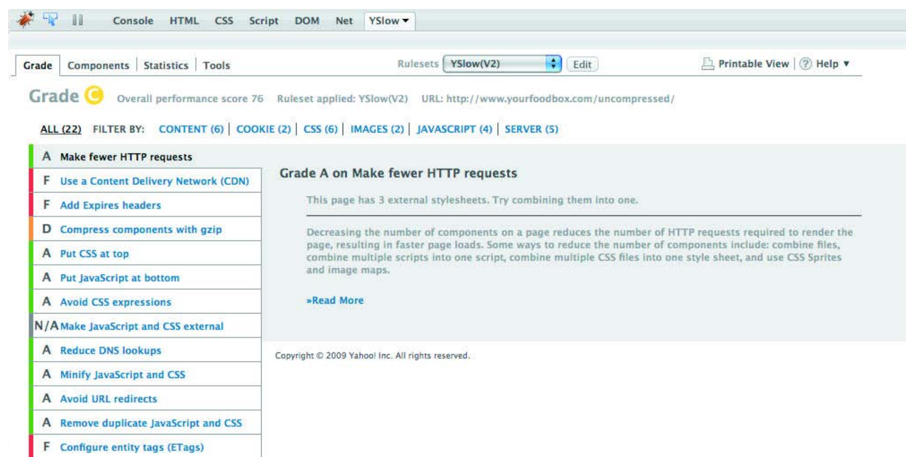


图 20-1　YSlow 找到了页面上的几个问题，我们需要解决它们

20.3 解决性能问题

你通过测试得到了一些数据，现在是该采取行动的时候了。你想怎么解决这些潜在问题？而且同以往一样，还需要去平衡某种措施所带来的成本和收益，而本章介绍的所有解决方案都各有长短。

20.3.1 设置超时报头

如果页面上的图片很少变化，可以将内容的超时时间设在未来的某个时间，间隔长达几个小时都行。

如果你用的是 Apache 服务器并且服务器设置里面有 `mod_expires` 模块，那么这个模块里的代码就能帮助你将所有改动过的 JPEG、GIF 和 PNG 文件缓存到本地长达 1 小时。

```
ExpiresActive On
ExpiresDefault A0

<FilesMatch "\.(jpg|png|gif)$">
  ExpiresDefault A3600
</FilesMatch>
```

其中 ExpiresDefault 值将缓存过期时间设定成 3600，而 3600 之前的那个 A 则代表的是从第一次访问开始。

ExpiresDefault 的值是以秒为单位的。在上面的例子中，将过期时间设定在距上一次修改文件的一个小时之后。你也应该想到，如果这个服务器上的图片不是每小时刷新一次，那这个设置几乎一点用都没有。

你还可以写得更具体一些，像这样：

```
ExpiresActive On
ExpiresByType text/html "access plus 30 seconds"
ExpiresByType text/css "access plus 1 hour"
ExpiresByType text/javascript "access plus 1 hour"
ExpiresByType image/png "access plus 1 day"
ExpiresByType image/jpg "access plus 2 months"
ExpiresByType image/gif "access plus 1 year"
```

这个方法可以让你对基于 MIME 类型的报头做特殊的处理，而不仅仅是在报头后面添加信息，而且这个方法对后台脚本也是有好处的。

还可以设计个有超长缓存时间的报头，它会在很久以后才过期：

```
<FilesMatch "\.(jpg|png|gif|css|js)$">
  Expires A31536000
</FilesMatch>
```

这条规则规定了首次从服务器拿到图片、样式表和 JavaScript 脚本文件后，缓存多长时间才会过期。这条规则可以大大减小服务器上来自浏览器的请求数。

设立长期缓存时间也是有弊端的，那就是当你需要更新一个文件的时候，必须要给这个文件换一个名字，或是用查询字符串来指向文件，因为你无法强制用户去清除他们的缓存。所以如果网页是一个静态网页，那么我不建议在经常变更的文件上采取长缓存时间的方法。不过还好，大多数网络应用框架会帮你处理长缓存时间的报头，框架所用的方法也是在部署或者呈现页面的时候改变文件名。如果你用的网络框架没有这个功能，那么在更新文件的时候就要手动更新文件的名字。因为你无法让用户的缓存过期。

20.3.2 用ETag改进缓存

现代浏览器在报头中支持一种叫作实体标签（entity tag）的东西，通常又称作 ETag。当浏览

器请求一个页面的时候，它也会记录这个 URL 的 ETag 值。假如用户再次请求了同一个页面，浏览器会用这次 URL 的 ETag 散列值同之前那个作对比，如果这两次的 ETag 是一样的，那么浏览器就不会去下载那个页面，而会从缓存中读取内容。这不但减小了带宽使用，而且还改进了用户体验。

ETag 可以根据页面大小、最新改动时间、校验码或是其他任何元素来计算，具体的计算方法完全取决于服务器。如果你部署了一个服务器端的框架，那么也可以用自己的方式来生成 ETag，这个方法对那些 RSS 订阅内容、动态 CSS 和动态 JavaScript 比较有价值，因为这些内容不会有“最新修改日期”这个值。

但是错误的 ETag 生成方法将导致用户每次访问的时候都会下载完整的页面，这种错误导致的最坏后果是用户根本无法看到页面上的新内容。如果你还为页面部署了负载均衡服务器，那么它背后的所有服务器可能会生成各自不同的 ETag。针对同一个页面，Apache 和 IIS 可能会生成不同的 ETag，所以在处理好 ETag 生成规则之前，你可能需要稍后再进行均衡服务器的部署。

如果用的是 Apache 服务器，那么在.htaccess 文件里加上下面这一行，就可以设定好 ETag 报头了：

```
FileETag MTime Size
```

这行设定告诉服务器，用“最新修改时间”和“文件大小”来生成 ETag。

1．在什么情况下应该用到ETag

像 RSS 订阅、网络服务（Web Service）或者是 blog 这类东西，它们并不会往硬盘里写数据，所以也没有所谓的“最新修改时间”可以利用。在这种情况下，你就需要在脚本中用到 ETag。如果你用的是服务器集群来承载网站，那么相同文件的“最新修改时间”会不一样，这样就会导致客户端认为某些文件没有被缓存，因为相同的页面由于不同的修改时间而生成了不同的 ETag。

如果无法保证“最新修改时间”的值是准确的，或者你根本拿不到这个值，那么你可能需要建立一套自己的机制来生成 ETag，这个机制可以是内容的散列表，或者是其他特殊机制。

如果网站后面有一套缓存机制，那么 ETag 会非常有用。前端的机器可根据 ETag 来决定是从后端服务器里获取内容，还是将自己本机缓存的内容发送给客户端。

2．禁用ETag

对于静态网站，特别是那些用到报头中 `expires` 属性的网站，ETag 并不十分适用，甚至还会减慢页面载入速度。因为只要用到了 ETag，客户端就会发送请求给服务器，虽然服务器回应的内容并不是完整的页面，但是这个活动仍然增加了流量。

在目前的情况下，还是把 ETags 禁用掉，然后完全依赖 expires 属性。在.htaccess 文件中加入下面这一行：

```
FileETag None
```

如果你没有忽略 ETag，并且将它设置得很完善，YSlow 会给你非常高的分数。可能到最后，你的网站内容都是建立在数据库上的，但是又不希望每次接到请求时都需要程序从数据库取数据，在这种情况下，你可能就需要生成自己的 ETag 或者是 expires 属性，然后让代码根据这些值来判断是不是要传送新的内容。

20.3.3 用资源服务器分发请求

对于那些特别大的网站，用相对路径来链接图片、样式表和脚本文件并不就是最好的方法。很多浏览器都会限制连接到同一个服务器的并发连接数量。如果网页上有 20 个资源文件，那么 IE 7 就会向服务器发起 20 个连接请求，而每次只能建立 2 个连接。这无疑拖慢了页面的速度。有些浏览器可能会允许多一些的并发连接，但是这也会增加服务器负载。为了能让页面尽快加载，最好的方法是将资源文件置于不同的服务器上。先看看下面的例子：

```
<img src="images/banner.jpg"/>
<script src="scripts/prototype.js"></script>
```

而更好的方法可能会是这种：

```
<img src="http://images.foodbox.com/banner.jpg"/>
<script src="http://scripts.foodbox.com/js/prototype.js"></script>
```

看上去，第二种方法似乎有些缺陷。首先你需要找到并维护更多的服务器，其次它调用了外部资源，最后你还需要记住一长串绝对链接，包括链接用的协议，是 HTTP 还是 HTTPS。如果你做的是电子商务网站，那么网页应该是在 SSL 下传输的，这样一来页面内所有的外部资源都要用 SSL 来链接。这意味着资源服务器也要开启 SSL 服务，然后再把 http://改成 https://。如果不这样做，用户可能会收到安全警告，或是影响有些内容的正常显示。

你还需要考虑这个方法是否值得。除了额外的服务器（更别提部署服务器资源所消耗的人力），还要考虑客户端服务对资源文件的缓存问题：通常这些文件都是缓存在客户端的，所以它们并不会每次都去下载同样的样式表或者图片。这种分发的方法更加适用于那些高流量的网站，特别是对于这些网站的初次访问用户来说，提升会更加明显。这个方法也会减小应用服务器的压力，因为你可以指定某台服务器用来专门供给图片。

将这些资源服务器推向云端是目前比较受欢迎的做法，像 Amazon S3 服务之类的。如果有类似的需求，那么我建议你去了解一下这方面的内容。

20.3.4　文件压缩

现代网络服务器可以提供压缩过后的内容，这可以缩短下载时间。如果你用的是 Apache 服务器，那么在.htaccess 里激活 `mod_deflate` 属性，并添加几条规则，就可以启动压缩功能：

```
AddOutputFilterByType DEFLATE text/html text/css \
    application/x-javascript
BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
```

头两个 `BrowserMatch` 规则的意思是不要向 Netscape 的老版本发送压缩过的内容，最后那条规则则针对伪装成 Netscape 的 IE 重新开启压缩功能。

20.3.5　压缩脚本文件

如果网站依赖体积很大的 JavaScript 和 CSS 库，那可能需要压缩它们的体积——可以通过删除注释、换行和空格的方式来减小文件体积。你还可以通过缩短变量和函数名的方法来压缩 JavaScript，同时减少文字的数量。

可能你会觉得这没什么，特别是已经通过服务器来提供压缩后的内容了。然而，通过去掉文件里的很多注释和空格，文件体积其实可以大大减小，而压缩工具并不能压缩掉如此之多的文字。

雅虎提供了 YUI 压缩工具[①]，这是一个基于命令行的组件，可以压缩 JavaScript 和 CSS 的体积，使用起来也很简单：

```
java -jar yuicompressor-2.4.2.jar \
     --type js prototype.js > prototype.min.js
```

你可以在开发环境下保持样式表和脚本文件的格式，然后复制这些文件，并用 YUI 工具压缩，将压缩后的文件部署到生产环境中就完工了。

体积压缩和自动化部署

专业的网络开发人员会把部署过程自动化。因为人工上传过程太枯燥了，而且开发人员需要不停地做这件事情。将部署过程自动化后，只需一次布置，就能让机器自己处理相关事务了，而且自动化可以确保所有的文件和步骤不被遗漏。

如果你的系统里已经有了自动化部署，往里面加入压缩文件的步骤应该不是很困难。你甚至都不需要很复杂的框架，单单是一个简单的 Ruby 脚本就能压缩 CSS 和 JavaScript 文件体积了，

---

① http://developer.yahoo.com/yui/compressor/。

然后还能将它们上传到网络服务器上。具体脚本在下面，注意这个脚本需要 get-scp[①]包（gem）[②]。你要做的只是设置好远程服务器，然后调用本地的 YUI 压缩工具就行。我建议将所有 YUI 压缩工具的 JAR 文件放置在一个 bin 文件夹里，然后将这个文件夹放在项目目录下：

performance/deploy.rb

```
# Scans your project for CSS and JS files and
# runs them through the Yahoo Compression utility
# and then uploads the entire site to your web server via SCP.

# Configure your settings below and be sure to supply the proper path
# to the Yahoo compressor. Set the COMPRESS flag to false to skip compression
COMPRESS = true
WORKING_DIR = "working"
REMOTE_USER = "homer"
REMOTE_HOST = "yourfoodbox.com"
REMOTE_PORT = 22

REMOTE_DIR = "/home/#{REMOTE_USER}/yourfoodbox.com/"

FILES = ["index.html",
         ".htaccess",
         "global_append.php",
         "global_prepend.php",
         "favicon.ico",
         "stylesheets",
         "images"
        ]

COMPRESSOR_CMD = 'java -jar bin/yuicompressor-2.4.2.jar'
# DONE CONFIGURING

require 'rubygems'
require 'net/scp'
require 'fileutils'

@errors = []

FileUtils.rm_rf WORKING_DIR
FileUtils.mkdir WORKING_DIR
FILES.each do |f|
  if File.directory?(f)
    FileUtils.cp_r f, WORKING_DIR
  else
    FileUtils.cp f, WORKING_DIR
```

① 用 sudo gem install net-scp 命令来安装 net-scp。

② gem 是一种包管理格式，作用是让 RubyGems 为 Ruby 语言提供库。——译者注

```
    end
  end

  # Upload files in our working directory to the server
  def upload(files)
    Net::SCP.start(REMOTE_HOST, REMOTE_USER, :port => REMOTE_PORT) do |scp|
     files.each do |file|
       puts "uploading #{file}"
       if File.directory?(file)
         scp.upload! "working/#{file}",  REMOTE_DIR, :recursive => true
       else
         scp.upload! "working/#{file}",  REMOTE_DIR
       end
     end
   end
  end

  # Minify all CSS and JS files found within the working
  # directory
  def minify(working_dir)
    files = Dir.glob("#{working_dir}/**/*.{css, js}")

    files.each do |file|
      type = File.extname(file) == ".css" ? "css" : "js"
      newfile = file.gsub(".#{type}", ".new.#{type}")
      puts "minifying #{file}"
      `#{COMPRESSOR_CMD} --type #{type} #{file} > #{newfile}`

      if File.size(newfile) > 0
        FileUtils.cp newfile, file
      else
        @errors << "Unable to process #{file}."
      end

    end
  end

  minify(WORKING_DIR) if COMPRESS

  if @errors.length == 0
    puts "Deploying"
    upload(FILES)
  else
    puts "Unable to deploy."
    @errors.each{|e| puts e}
  end
```

你还可以改造这些代码，比如说加一个功能，把所有的 CSS 文件合并成一个新文件，这样你就能把 HTML 中调用 CSS 的链接减少到只有一个，并只调用新的链接。这样在压缩时就只有

一个文件会被压缩，而且会减少客户端对服务器发出的请求数目。

20.4 图片优化

在第 10 章中，你已经学会了用 Photoshop 来优化图片。除此之外，其实还可以进一步优化图片。比如像雅虎的 Smush-It!服务就会利用几个开源工具帮你优化图片，然后 YSlow 插件可以将页面里的图像文件发送到这个服务器去优化。我在 Smush-It!里运行 Foodbox，在图 20-2 里可以看到结果。

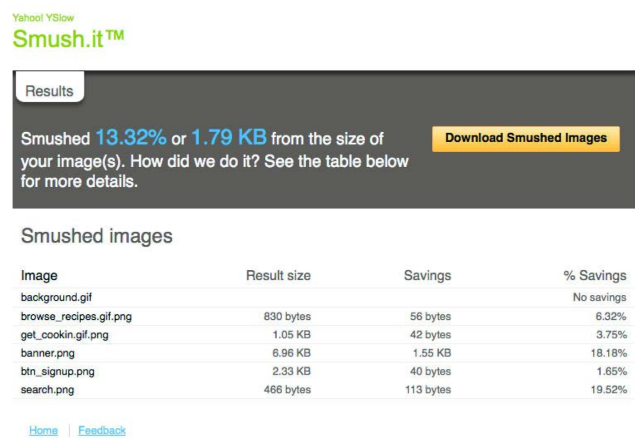


图 20-2 Smush-It!将图片压缩了 13%的体积，差不多比 2KB 稍微小一点

报告显示，Smush-It!可以将图片缩小 13%。乍听起来，这个成绩还不错。再想一想，这只不过为我们节省了大约 2KB 的空间。更糟糕的是，这个方法还需要将一些 GIF 文件转换为 PNG 文件，这意味着如果在页面上使用了这些文件，那么需要修改样式表和标记文件。但是在目前的情况下，我还是希望这些东西保持原样。

自己动手

Smush-It 使用了 Pngcrush[①]作为优化 PNG 的工具，然后用 ImageMagick[②]来确定图片格式，并把 GIF 转化为 PNG，最后用 JPEGTran[③]来去掉 JPEG 文件里的 `meta` 信息。

我会用 get_cookin.gif 来做个小实验，将它转换为 PNG，用 Pngcrush 优化一下：

① http://pmt.sourceforge.net/pngcrush/。
② http://www.imagemagick.org/script/index.php。
③ http://sylvana.net/jpegcrop/jpegtran/。

```
convert get_cookin.gif tmp.png
pngcrush -rem alla -reduce --brute tmp.png get_cookin.png
```

比较两个文件后发现，它们的体积几乎是一样的。所以在这个例子里，这些多出来的步骤几乎没有必要，用 Photoshop 压缩已经足够了。

但是，如果换成注册按钮：

```
pngcrush -rem alla -reduce --brute btn_signup.png btn_signup2.png
```

体积从原来的 2.4 KB 降到了 2.3 KB，虽然体积有了些许变化，但是效果也不是很明显。

所以在这个例子中，进一步优化图像并不是非常必要，但是在建造你自己网站的时候，这一点还是需要考虑的。如果它对你有利，那么记住要在自动部署的代码里加上这一段。

20.5 小结

在实现了本章讨论的技巧之后，用户会在第一次访问网站之后感觉到很大的速度提升（参见图 20-3）。这里只是粗略地描绘了一下性能优化的图景，你也应该对相关的知识有了比较深入的了解，当你在这方面遇到困难的时候，应该知道往什么方向努力。对 ETag、压缩、减小体积和 `expires` 报头的恰当设置可以很好地帮助你减少请求数量，从而大大节省带宽使用量。

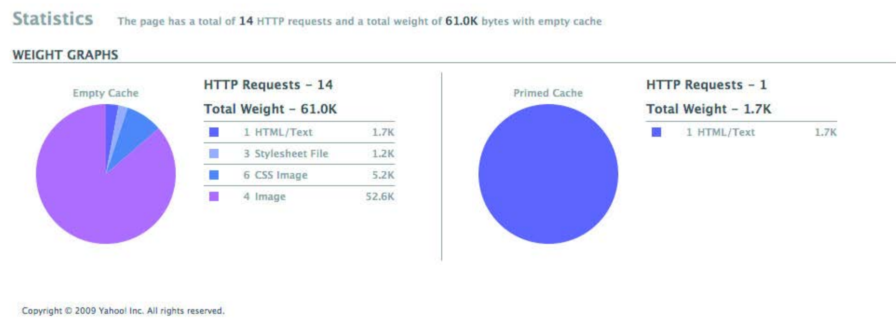


图 20-3 经过优化之后，反复返回访问网站的用户不会多次请求服务器

第 21 章 后续工作

建造 Foodbox 的工作已经完成，现在你可能正盯着它寻思：接下来还能做什么呢？或者在想：如果以后我要做点儿不一样的东西，应该怎么办？本章会介绍一些方法，你可以研究一下，这对你以后的开发或许会有好处。

21.1 其他页面和模板

我们用了一整本书来讲设计一个网站，但实际上只完成了一个页面。通常一个网站只有一个页面是不够的，同时其他的页面跟首页应该是不同的。

二级页面

一个网站有两套设计是很常见的，通常首页的设计是一套，然后其他的次级页面共用另外一套的风格。这套设计跟首页的设计非常类似，但一般 Banner 更小，导航和内容都会修改。这套二级页面模板通常是用来组织内容的。下面会用很快的速度创建一个二级模板，并加以使用。

1. 创建二级页面模板

复制 index.html 文件到一个名叫 level2.html 的新文件，定位到下面的这个标签，并删除其中的所有内容：

```
<div id="middle">
  ...
</div>
```

然后将内容替换为：

final/level2.html

```
<div id="middle">
  <div id="leftcol">
  </div>
```

```
  <div id="rightcol">
  </div>

</div> <!-- end of middle container -->
```

然后需要将 `class="level2"`应用到 `body` 标签上。可以利用这个新的类，将 CSS 选择器的范围限定到二级页面模板上。

在布局上将页头缩小到 54 像素高，然后用到之前学过的 `float` 技巧来布局双栏排版：

final/stylesheets/layout.css

```
.level2 #header{height:54px;}
#middle {width:100%;}
#leftcol, #rightcol{
  margin:18px;
  float:left;
  display:inline;
}
#leftcol{width:558px;}
#rightcol{width:270px;}
```

因为缩小了页头，所以需要新的 Logo 和背景图片。这个步骤可以用 Photoshop 来完成，但是如果用 ImageMagick[①]会更快。打开终端窗口或者是命令行提示，进入图片目录。

将新 Banner 改为 36 像素高：

```
convert -geometry x36 banner.png banner_small.png
```

在 level2.html 模板中将 Banner.png 改成 Banner_small.png。

因为页头是 54 像素高，所以背景图片应该也是这个高度。可以用 ImageMagick 的 `crop`（剪切）命令：

```
convert -crop 1x54+0+0 background.gif background_level2.gif
```

`crop` 命令的几个参数分别是图像的宽、高和起始坐标。“`0+0`”规定了剪切从图片的最左上角开始。

为了给页面加上样式，需要修改中间区域的背景颜色和背景图片：

final/stylesheets/style.css

```
.level2 #middle{background-color:#fff8e4}
body.level2 {background: #fff url('../images/background_level2.gif') repeat-x;
}
```

你看，其实创建一个模板只需要这么几个简单的步骤。

---

① 可以在 http://www.imagemagick.org/script/index.php 下载 ImageMagick。如果你是 Mac 用户，可以通过 MacPorts 来安装它；如果你是 Linux 用户，包管理程序中可能会包括它。

2．用模板打造一个登录页面

你可以用 level2.html 模板创建登录页面。将刚刚完成的 level2.html 复制到一个叫做 login.html 的新文件，在 login.html 中加上如下代码：

final/login.html

```
<div id="leftcol">
  <h2>Log in</h2>

  <form id="login" method="post" action="/user_sessions">
  <table>
    <tr>
      <th><label for="username">Username</label></th>
      <td>
        <input type="text" name="username"
               id="username" class="text">
      </td>
    </tr>
    <tr>
      <th><label for="password">Password</label></th>
      <td>
        <input type="password" name="password"
               id="password" class="password">
      </td>
    </tr>
    <tr>
      <th> </th>
      <td>
        <input type="checkbox" name="remember" id="remember" class="checkbox">
        <label for="remember">Remember me</label></td>
      </th>
    </tr>
    <tr>
      <th> </th>
      <td><input type="submit" value="Log in"></td>
    </tr>
   </table>
  </form>
</div>
<div id="rightcol">

  <h2>Already have an account?</h2>
    <a href="/signup/">
      <img src="images/btn_signup.png" alt="Sign up">
    </a>
</div>
```

你还需要在 layout.css 中加入如下代码：

final/stylesheets/layout.css

```
form {margin-left:36px;}
form table{border:0px;}
form table tr{height:36px;}
```

完成以上步骤之后，页面应该看起来跟图 21-1 差不多。



图 21-1　用二级页面模板打造的简单登录页面

目前为止，我们进行得很顺利，但是这个工作流的扩展性对于那些有上百个页面的网站来说是远远不够的。

21.2　高级模板

如果不断地用复制模板的方式来新建页面，那么你马上会坠入维护的噩梦。虽然在 CSS 里面可以处理所有的颜色，但是如果遇到要修改页脚里链接的情况，你该怎么办？如果用这种复制粘贴的方式做了 20 个网页，那么你就需要手动管理这 20 个页面里面的链接和内容。如果你打算维护一个有层级的链接，那么那些连接到外部文档的链接会变得更难管理。

如果你打算开发一个静态网站，那么可以用 Dreamweaver 来跟踪模板。把页面同某个模板关联起来，如果改动模板里的某个链接，Dreamweaver 会自动帮你更新与之关联的页面。如果你用了 ColdFusion 或是 PHP，Dreamweaver 会是一个很棒的工具，因为它能自动追踪页面上的链接和图片。如果把一个页面移动到另外一个文件夹里，Dreamweaver 会自动帮你更新页面里样式表、图片和其他所有文件的相对链接。但是这个工具的价格偏高，而且有些时候会显得功能冗余。因此，Dreamweaver 并不是维护静态网站的唯一选择。

StaticMatic[1]和 Nanoc[2]是两个简单且灵活的静态网站管理工具。这两个工具都是用 Ruby 写

① http://staticmatic.rubyforge.org。
② http://nanoc.stoneship.org/。

的，用起来非常简单，都有详尽的文档，而且它们最大的优点是——免费。

现在，大多数网站都不是静态的。但把模板转换成某种框架或者某种语言的可用版本，应该也不是什么难事，你可以用 PHP、ColdFusion、Ruby on Rails、Django、Perl、.NET 或者其他任何基于 Web 的框架来做这件事情。

多数现代 Web 框架都有内建的模板机制，所以将设计转换成模板不难，难的是怎么设计。

21.3　网格系统和 CSS 框架

在本书中，我有意没有引导你使用那些流行的 CSS 框架，因为我希望你可以创建自己的网格。但是现在你已经知道网格系统是怎么一回事了，那么可以去看看一些开源的布局框架了。

21.3.1　YUI网格

YUI（Yahoo! User Interface Libray，雅虎用户界面库）内置了一个网格建造器，叫做 YUI 网格[①]，这个建造器让创建网格变得易如反掌。用它生成的一个简单的 Foodbox 模板看起来是这样的：

final/yui_foodbox.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <title>YUI Base Page</title>
   <link rel="stylesheet"
 href="http://yui.yahooapis.com/2.7.0/build/reset-fonts-grids/reset-fonts-grids.css"
 type="text/css">
</head>
<body>
<div id="doc2" class="yui-t3">
   <div id="hd" role="banner"><h1>Foodbox</h1></div>
   <div id="bd" role="main">
        <div id="yui-main">
        <div class="yui-b"><div role="main" class="yui-g">
         <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
       sed do eiusmod tempor incididunt ut labore et dolore
       magna aliqua. Ut enim ad minim veniam, quis nostrud
       exercitation ullamco laboris nisi ut aliquip ex ea commodo
       consequat. Duis aute irure dolor in reprehenderit in voluptate
       velit esse cillum dolore eu fugiat nulla pariatur.
    </p>
  </div>
```

① http://developer.yahoo.com/yui/grids/builder/。

```
</div>
        </div>
        <div role="search" class="yui-b">
         <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
      sed do eiusmod tempor incididunt ut labore et dolore
      magna aliqua. Ut enim ad minim veniam, quis nostrud
      exercitation ullamco laboris nisi ut aliquip ex ea commodo
      consequat. Duis aute irure dolor in reprehenderit in voluptate
      velit esse cillum dolore eu fugiat nulla pariatur.
    </p>
    <p>
      Excepteur sint occaecat cupidatat non proident,
      sunt in culpa qui officia deserunt mollit anim id est laborum.
          </p>
        </div>

        </div>
     <div id="ft" role="contentinfo"><p>Copyright 2010 Foodbox</p></div>
  </div>
  </body>
  </html>
```

你可以用这个建造器为页面添加样式，只需要在框架内写代码就好了。只要用对了 id 和 class，整个过程会很顺利。在上面那个例子里面，最外层的 div 指定了 960 像素宽，它的 id 是 doc2。另外 yui-t3 这个类规定了左栏的宽是 300 像素。这个工具有很多的设置项，具体请看相关文档。[①]

上面例子中的版本还重置了所有元素的格式（参见图 21-2），你可以看到 Foodbox 的标题也变得非常的小。在应用了 YUI 之后，你需要再添加自己的样式表，为标题类元素加上样式。

图 21-2　Foodbox 的 YUI 网格布局，没有加任何装饰

21.3.2　960 网格系统

广受欢迎的 960 网格系统[②]是 YUI 之外的另一个选择。这个系统将页面预置成 960 像素宽，

---

① http://developer.yahoo.com/yui/grids/。

② http://960.gs/。

并将之分成 12 列或者 16 列。当你以此为基础打造页面布局的时候，系统会让你用 class 来确定每个区域需要占几列。

如果选用了 12 列的网格，那么你需要将页头和页脚设置为 12 列宽，侧边栏的宽度为 4 列，这样主区域的宽度就是 8 列了。列与列之间的间隔就不用考虑了，960 网格系统会帮你自动配置好。

代码如下：

final/960_foodbox.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>960 Grid System &mdash; Demo</title>
<link rel="stylesheet" href="http://960.gs/css/reset.css" />
<link rel="stylesheet" href="http://960.gs/css/text.css" />
<link rel="stylesheet" href="http://960.gs/css/960.css" />
</head>
<body>

  <div class="container_12">

    <div id="header" class="grid_12">
      <h1>Foodbox</h1>
        </div>


        <div id="sidebar" class="grid_4">
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
         sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
         Ut enim ad minim veniam, quis nostrud exercitation ullamco
         laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
         in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
         pariatur.
      </p>
      <p>Excepteur sint occaecat cupidatat non proident, sunt
        in culpa qui officia deserunt mollit anim id est laborum.
      </p>

        </div>

    <div id="main" class="grid_8">
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit
         sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
         Ut enim ad minim veniam, quis nostrud exercitation ullamco
         laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
         in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
```

```
        pariatur.
      </p>
      <p>Excepteur sint occaecat cupidatat non proident, sunt
        in culpa qui officia deserunt mollit anim id est laborum.
      </p>
    </div>
    <div id="footer" class="grid_12">
      <p>Copyright &copy; 2010 Foodbox</p>
        </div>

  </div>

</body>
```

这个网格可以帮你用几行代码就搭建好一个简单的布局（参见图 21-3）。960 网格系统使用了 13 像素的字体和 1.5 的行高——这里的 1.5 是相对值，指的是行高为字体的 1.5 倍。因此现在行高就是 19.5 像素，这对剪切图片造成了困难，所以你可能需要在自己的样式表里面改动字体大小和行高。

图 21-3　用 960 网格系统生成的无样式 Foodbox

如果你想把 960 网格系统融合到自己的工作流里去，可以上它的网站下载一份可以打印的文档，Photoshop 模板对你设计自己的网站也很有帮助。

21.3.3　框架不是万能的

虽然这些框架让基于网格的布局设计变得简单异常，但是掌握一些关于基线网格的知识仍然是有必要的，特别是当你选择字体、字号和放置图片的时候，这些知识会很有用。另外，这些系统通常都包含很多你并不需要的 CSS 代码。960 网格系统还需要你关联很多文件，这可能会降低性能。如果你准备使用这些框架，一定要处理好压缩文件体积和缓存技术。

完全了解其原理，然后再使用它们，这样你才能从 CSS 框架中获益。

21.4 替换 CSS

作为程序员，跟 CSS 打交道可能会比较头疼，也许还会显得有点儿多余。CSS 并没有变量和继承的概念，所以你到最后可能需要把同一段代码写很多遍。

下面这段代码带来的将是维护噩梦：

```
#latest_recipes{
   clear:both;
   margin: 18px 18px 0 18px;
}

#latest_recipes h3{
  margin-left:18px;
}

#latest_recipes p{
  margin-left:36px;
}
```

为了缩小范围而反复使用 `latest_recipes`，这看起来就不太对。

现在有一些开源的项目制作了一些 CSS 生成器，它们自己有一套标记语言，用这些标记语言写完代码以后，生成器会将之转换为静态的 CSS 文件，然后就可以在应用中使用这些 CSS 了。接下来我会介绍 Less，它是一个简单但却强大的 CSS 预处理器，它背后的实现语言是 Ruby。[①]

Less CSS

通过 Ruby，Less[②]可以完全发挥你的 CSS 水平，很轻松地写出 CSS 样式表。

通过 Less，可以将上面那些代码有逻辑地整理成下面这样：

final/less_examples.less

```
#latest_recipes{
   clear:both;
   margin: 18px 18px 0 18px;

   h3{
    margin-left:18px;
   }

  p{
```

① 另外，Sass 可能也会唤起你的兴趣，Sass 与 Less 类似，但语法略难一些。你可以在 http://sass-lang.com/了解更多 Sass 的知识。

② http://lesscss.org/。

```
    margin-left:36px;
  }
}
```

这个文件被转化为 CSS 文件时，h3 以及 p 标签选择器的范围会被自动配好。

Less 的真正强大之处则在于对变量和表达式的支持。这种支持可以让你写出下面这种代码：

final/less_examples.less

```
@text_color: #fff;
@width: 900px;
@font_size: 12px;
@line_height: @font_size * 1.5;
@margin: @line_height;
@sidebar: @width / 3;
@main: @width - @sidebar - @margin;

body {color: @text_color; }
#page { width: @width; margin: 0 auto; }
#middle { width: @width; }
#main { width: @main; }
#sidebar { width: @sidebar; }
```

虽然浏览器并不理解这种形式的样式表，但是通过 Less 预处理器的解析和转换，你就能将之变成标准的 CSS 文件：

```
less source/style.less stylesheets/style.css --watch
```

--watch 命令监听对源文件的改动，当你保存这些文件的时候，Less 会自动生成 CSS 文件。这不仅让测试变得异常轻松，还会大大简化样式表的管理。另外，你还可以轻松地把这个过程融合到自动化部署的过程中去。

21.5　不要忘记为有版权的照片付钱

在本书里用到的那个意大利面的 Logo 来自于 iStockphoto[①]。在书中的例子里面用的都是带水印的版本。但是，如果网站要上线，那就必须去买最终版本[②]。这个听起来应该属于常识性的东西，但是如果你去 Photoshop Distasters[③]这个博客看一看就知道，很多人在发布网站的时候使用的是带水印的图片——甚至有人出版印刷品的时候也是这样！所以，千万不要忘记为那些带版权的照片（或是其他任何你使用了的资源）付费。另外要记住，不要因为一张图片在 Google Image 中出现了，就以为你可以免费使用它。

---

① http://www.istockphoto.com。

② 为了将之用作本书的例子，我已经购买了一次，所以在完成那些例子的时候，你可以使用这个图片。

③ http://www.photoshopdisasters.com/。

21.6 视觉效果

以前，那些淡入淡出之类的动画必须使用 Flash 才可以实现。但是现在，很多类似的效果都可以借用像 jQuery、Prototype 和 Scriptaculous 这样的 JavaScript 库来实现。这些开源的 JavaScript 框架旨在简化对 HTML 元素的控制、动画效果的制作和 Ajax 的实现。现在就用 jQuery 来为页面上的意大利面的图片加入一个淡入淡出效果吧。

首先需要找一些大小合适的图片来实现这个效果。这些图片应该有 594 像素宽、144 像素高，这样才能放到页面的那个区域里去。我打算使用三张 Flickr 上的图片，它们的版权许可都是“Creative Commons Attribution”[①]——这意味这在使用照片的时候需要注明原作者。我选中了这样的三张图片[②]：

- http://www.flickr.com/photos/pencapchew/3108612635/
- http://www.flickr.com/photos/stevendepolo/3523644703/
- http://www.flickr.com/photos/denniswong/3486409564/

21.6.1 缩放图片

图片需要被缩放到 594 像素宽，这样它们就能像意大利面的图片一样放进那一块区域了。ImageMagick 的 Geometry 选项能在只提供图片宽参数的情况下，保持图片不变形（这段代码限制了宽度和高度的比例）：

```
$ convert -geometry 594x originals/tacosalad.jpg tacosalad.jpg
$ convert -geometry 594x originals/phadthai.jpg phadthai.jpg
$ convert -geometry 594x originals/chickenmac.jpg chickenmac.jpg
```

接下来需要做的是裁剪，只需要将图片正中裁剪下来即可。先用命令看看图片的高度：

```
$ identify tacosalad.jpg
tacosalad.jpg JPEG 594x394 594x394+0+0 8-bit DirectClass 146kb
```

从数据中来看，图片高度是 394 像素，用这个高度减去 144 然后除以 2。最后算出来需要开始从 9 像素 × 125 像素的地方开始裁剪：

```
$ convert -crop 594x144+0+125 tacosalad.jpg tacosalad.jpg
$ convert -crop 594x144+0+125 phadthai.jpg phadthai.jpg
$ convert -crop 594x144+0+125 chickenmac.jpg chickenmac.jpg
```

21.6.2 写代码

为了实现淡入淡出的效果，需要在载入下一张图片的时候让当前的图片消失。Flash 这类软

---

① 这是一种版权协议，允许对作品进行复制、分发、改造和合成，但是要在使用的时候注明原作者。——译者注

② 如果图片已经无法访问，那么请从本书附带的源代码中获取。

件用时间轴来控制效果。每个图片都有自己的图层，然后等到要淡入的时候，就将两个图层在时间轴上交叠。JavaScript 没有时间轴，但它也为我们提供了做出这种效果的方法。

从图片数组开始。就本例而言，我们尽量简化问题，不修改淡入淡出中的默认文本。

final/javascripts/crossfade.js

```
images = [
"images/pasta.jpg",
"images/tacosalad.jpg",
"images/phadthai.jpg",
"images/chickenmac.jpg"
]
```

这个数组里图片的顺序是很重要的，这决定了图片将以什么样的顺序出现。

然后需要将它放到页面中，一张一张地摞起来。整个淡入淡出的代码会让最上面的那个图片慢慢消失，显露出第二张图片。然后每 5 秒重复一次，当到达最后一张图片的时候，程序会把状态重置。

我们会用 CSS 和 JavaScript 在 Banner 里把图片摞起来：

final/javascripts/crossfade.js

```
Line 1  var image_box = $("#main_image");
     2
     3  image_box.css({'position' : 'relative', 'height': '144px'} );
     4  var image_html = "";
     5  for(var i = 0; i < images.length; i++) {
     6    image_html += '<img style="position:absolute;top:0; z-index:' +
     7    (images.length - i) + '" src="' + images[i] + '" id="image_' + i + '"/>';
     8  };
     9
    10  image_box.html(image_html);
```

代码一开始就获取了图片的容器元素，然后用 JavaScript 将这个容器的 `position` 设置为 `relative`，并给它赋值一个具体的高度，以防止它塌陷而不能显示任何图片。接下来新建了一个字符串，用来存储图片的 HTML 代码，然后遍历所有的图片为它们创建 HTML 代码。在这里面又用到了绝对定位来让图片跟容器的左上角对齐，并且用到了 CSS 里的 `top`、`left` 和 `z-index` 属性。这里用的都是内联样式，[①]`top` 和 `left` 属性的值都是 0，`z-index` 则是摞起图片的顺序，它的值就是图片数组的索引值，这样图片就一个叠一个地摞起来了。

每个图片都有自己的 `id`，格式是 `image_i`，`i` 是图片在数组里的索引值。这个 `id` 我们在一

① 之前我说过内联样式是不好的做法，但是现在这种样式又是必需的，虽然像 `postion`、`left`、`top` 这种属性可以在 CSS 中完成设置并赋予图片的公共类，但是 `z-index` 这个值是每个图片都不一样的。所以在这里我用了内联样式。

会儿指定图片淡出的时候就会用到。

把图片的 HTML 写好之后，就可以用容器的 html()方法来把它写进去（参见代码第 10 行）。

需要初始化一个 counter 变量来跟踪图片栈里的图片，另外图片过渡每 5 秒进行一次，所以要用到 setInterval()方法，这个方法可以在给定间隔内调用一个方法。

final/javascripts/crossfade.js

```
Line 1  var i = 0;
     2  var delay_in_miliseconds = 5000;
     3
     4  setInterval(function(){
     5    $("#image_" + i).animate({ opacity: 0}, 3000);
     6    i++;
     7    if(i == images.length) i = 0;
     8    $("#image_" + i).animate({ opacity: 1}, 3000);
     9  },delay_in_miliseconds);
```

第 4 行里，向 setInterval()方法传递了一个匿名函数。这个函数用计数器（counter 变量）来判定需要淡出的图片，然后 counter 变量自增 1，这样就得到了要显示的下一张图片。但是在激活下一张图片之前，先要检查索引是不是超过了图片的总数，如果是，将 counter 变量重置为 0。最终，就像洗牌一样，一张张地显示图片，一张张地淡出。

一个相对区域里的绝对定位

通常在你使用坐标定位某个元素的时候，该元素的坐标是相对于浏览器窗口左上角的。所以，如果你想让某个元素出现在浏览器左上角往下 100 像素往左 18 像素的地方，应该这样写：

```
.box{
  position:absolute;
  top:100px;
  left:18px;
}
```

但是如果把某个元素的 position 设置为 relative，该元素就成了绝对定位其子元素的参照元素。这样，打造图片栈的过程就容易了很多，只需要将相对于图片容器左边的距离都设为 0 就可以了。

21.6.3 把动画放到主页上

只有调用了上面那段代码，谈入淡出脚本才会起作用。因此需要在主页上引入 jQuery 库和脚本。

打开首页代码，在 body 标签关闭之前加上下面的代码：

final/index.html

```
<script type="text/javascript"
   charset="utf-8"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js">
</script>

<script type="text/javascript" charset="utf-8"
  src="javascripts/crossfade.js">
</script>
```

这两段代码加载了 jQuery 库和淡入淡出脚本。我还有个想法是，浏览器是根据原始 URL 来缓存文件的，那么通过在页面上加载来自同一个源的通用库（比如说 jQuery），应该可以提升页面的性能。因为如果用户从另外一个网站跳到网站，而我们都用了同一个源，那么有些库可能就已经被浏览器加载过了。Google 通过 Ajax 库的 API 提供了这样的服务[①]，可以从它那里加载 jQuery，这样用户在本地拥有跟我们相同的 jQuery 的几率就变大了。

最后，需要对主页图片的标签做小小的改动。代码会向一个 ID 为 main_image 的元素注入一些摞在一起的图片。而在首页上，main_image 这个 ID 是属于那张意大利面图片的。因此需要把这个 ID 从那个图片上去掉，然后在图片周围加一个 div，给这个 div 加一个叫做 main_image 的 ID：

final/index.html

```
<div id="main_image">
  <img src="images/pasta.jpg" alt="Pasta and marinara sauce">
</div>
```

这就弄好了！现在有了一个很简单的图片淡入淡出程序，而且这个程序几乎不引人注意。当 JavaScript 被禁用之后，人们还是可以看到那个意大利面图片。当使用 JavaScript 的时候，一定要保证 JavaScript 跟内容无关。另外也不要用 onclick 和 onmouseover 方法，因为将显示和互动混杂在一起是完全没有必要的。如果你没有提供一个替代方案，这会引起可访问性方面的问题。

隐式 JavaScript

隐式 JavaScript（unobtrusive JavaScript）指的是 JavaScript 逐渐地同内容完全分离。这种方法可以让你利用很简单的方式就能增强自己的网站，同时又不会影响为那些无法使用 JavaScript 的用户提供的功能。

① http://code.google.com/apis/ajaxlibs/documentation/。

> 通常，你可能会用以下方式让链接在新窗口中被打开：
>
> ```
> <a href="#" onclick="window.open('help.html');">Help</a>
> ```
>
> 但是如果用户没有 JavaScript，他就无法访问这个页面了。另外一个稍好一些的方法是，在 a 标签中使用 href 标明地址，然后用 JavaScript 监听这个链接，当它被单击的时候，脚本会截获 href 的值，然后在新窗口中把它打开。
>
> 隐式 JavaScript 在此之上又有些改进：复用性大大增强。假设你需要让页面上所有 popup 类的链接都在新窗口中被打开，下面这段简短的 jQuery 代码就能实现这个功能：
>
> ```
> (document).ready(function(){
>   var links = $("a.popup");
>
>   links.click(function(event){
>   event.preventDefault();
>   window.open($(this).attr('href'));
>   });
> });
> ```

如果想了解更多关于隐式 JavaScript 的内容，请访问 http://onlinetools.org/articles/unobtrusive-javascript/。

JavaScript 可以让页面生动起来，如果使用得当，每个用户的体验都会得以提升。记得让代码变得“隐式”，确保所有的功能在没有 JavaScript 时都能正常运行。最后，别忘记压缩脚本的体积！

21.7 多试多做

在本书中，用了一个比较保守的配色和设计方案。现在你已经经历了所有的过程，是时候从头开始，在页面中实现你自己的想法和设计了。起草自己的草图，创建 Logo，选择颜色和字体，然后用网格系统打造你的网站吧。多看其他的网站寻找灵感，用 Firebug 来查看那些网站是如何建造的；向别人学习，然后不断地实践、实践、再实践。跟编程一样，Web 设计也需要花费毕生精力才能精通。

所以，请一直学习、探索和尝试吧，你一定会从中得到乐趣。

第 22 章

推 荐 阅 读

程序员会通过大量地阅读书籍和浏览网站来保持他们的技能领先，这已经是一个公开的秘密了。本章我列出了一些自己觉得会对你有帮助的书，这些书基本上涵盖了我在自己的书中所讲到的方方面面，可以帮助你进一步探索这些知识。

22.1 色彩资源

色彩从来都是一个复杂的话题，对色彩的深入理解能帮助你将信息转化成可以抓住用户眼球的设计。以下几本书是我认为比较有价值的，你可以挑几本来扩充你自己的藏书。

- Albers，Josef．*The Interaction of Color* [Alb75]
- Itten，Johannes．*The Elements of Color* [Itt97b]
- Itten，Johannes．*The Art of Colour: The Subjective Experience and Objective Rationale of Color - Revised Edition* [Itt97a]
- Morton，Jill．*A Guide to Color Symbolism* [Mor97], Colorcom, 1998.

22.2 关于字体和排版的书

对字体和排版的深入理解也是一个成功的 Web 开发人员所必备的素质，所以我建议你继续深入了解这些东西。下面的这两本书写得很棒，它们向你展示了如何在自己的设计中有效地应用字体和网格系统：

- Ruder，Emil．*Typography* [Rud81]
- Muller-Brockmann．Josef，*Grid Systems in Graphic Design* [MB96]

22.3 技术书籍

本书为你开启了网络开发之路，如果想要顺着这条路走下去，下面这些书将会非常有用。

- Ash，Tim．*Landing Page Optimization The Definitive Guide to Testing and Tuning for Conversions* [Ash08]
- Clifton，Brian．*Advanced Web Metrics with Google Analytics* [Cli08]
- Goto，Kelly and Cotler，Emily．*Web ReDesign 2.0: Workflow that Works* [GC04]
- Krug，Steve．*Don't Make Me Think! A Common Sense Approach to Web Usability* [Kru04]
- Meyer，Eric.A．*Cascading Style Sheets: The Definitive Guide，Third Edition* [Mey06]
- Sydik，Jeremy．*Design Accessible Web Sites* [Syd08]
- Veen，Jeffrey．*The Art & Science of Web Design* [Vee00]
- Zeldman，Jeffrey．*Designing With Web Standards* [Zel06]

22.4 网站资源

互联网上有无穷无尽的指南和教程可以帮助你构建网站，下面是我发掘的一些跟本书相关的资源，我认为这些资源是我找到的很好的一些网络资源。

About Hearing Loss.. http://www.miracle-ear.com/abouthearingloss.aspx
Miracle-Ear 网站包括很多关于失聪患者的信息，包括类型和病因。

A List Apart: CSS @ Ten: The Next Big Thing .... http://www.alistapart.com/articles/cssatten
Håkon Wium Lie 讨论了在样式表中使用@font-face 的问题。

A List Apart: Going to Print..http://alistapart.com/articles/goingtoprint
Eric Meyer 写的一篇文章，讨论了使用 CSS 为网站提供适合打印的样式表。

Brandcurve: “Color Meanings Around the World”
.... http://www.brandcurve.com/color-meanings-around-the-world/
一个颜色列表以及其国际含义。

Lighthouse International – “Making Text Legible Designing for People with Partial Sight”
....http://www.lighthouse.org/accessibility/legible/
一些基本指南，关于选择适合所有人浏览网页的有效的、易辨认的文字。

Safalra:“The Myth of Web-Safe Fonts”
....http://safalra.com/web-design/typography/web-safe-fonts-myth/
提供了字体和 CSS 的基础知识。

Unit Interactive:“Better CSS Font Stacks”
....http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/
讨论了字体栈，提供了很多优秀的示例。

WebAim:“CSS in Action: Invisible Content Just for Screen Reader Users”
.... http://www.webaim.org/techniques/css/invisiblecontent/
关于如何向屏幕阅读器提供其他内容的说明和示例，比如，导航跳转链接。

附录 *A*

参 考 书 目

[Alb75] Josef Albers. *Interaction of Color*. Yale University Press, New Haven CT, 1975.

[Ash08] Tim Ash. *Landing Page Optimization: The Definitive Guide to Testing and Tuning for Conversions*. Sybex, New York, 2008.

[Cli08] Brian Clifton. *Advanced Web Metrics with Google Analytic*s.Sybex, New York, 2008.

[GC04] Kelly Goto and Emily Cotler. *Web ReDesign 2.0: Workflow that Works*. Peachpit Press, Berkeley, 2004.

[Itt97a] Johannes Itten. *The Art of Color: The Subjective Experienceand Objective Rationale of Color Revised Edition*. Wiley, New York, 1997.

[Itt97b] Johannes Itten. *The Elements of Color*. Wiley, New York,1997.

[Kru04] Steve Krug. *Don't Make Me Think! A Common Sense Approach to Web Usability*. Peachpit Press, New York, 2004.

[MB96] Josef Müller-Brockmann. *Grid Systems in Graphic Design*.Niggli, Sulgen, Switzerland, 1996.

[Mey06] Eric Meyer. *CSS: The Definitive Guide*. O'Reilly Media, Inc.,Sebastopol, CA, third edition, 2006.

[Mor97] Jill Morton. *A Guide to Color Symbolism*. Colorcom, Broomfield CO, 1997.

[Rud81] Emil Ruder. *Typography*. Niggli, Sulgen, Switzerland, 1981.

[Syd08] Jeremy Sydik. *Design Accessible Web Sites: 36 Keys to Creating Content for All Audiences and Platforms*. The Pragmatic Programmers, LLC, Raleigh, NC, and Dallas, TX, 2008.

[Vee00] Jeffrey Veen. *The Art and Science of Web Design*. New Riders Press, Upper Saddle River NJ, 2000.

[Zel06] Jeffrey Zeldman. *Designing Web Standards*. Peachpit Press, New York, second edition, 2006.

Web Design for Developers

A Programmer's Guide to Design Tools and Techniques

写给程序员的Web设计书

“真希望我在第一次做网站的时候就能看到这本书。本书涵盖了Web开发的方方面面，当你需要改进网站时，这本书能为你解答很多问题。”

——Shae Murphy，Social Brokerage CTO

“如果你已经准备好要踏入Web设计的奇妙世界，读这本书可以让你清晰且有效地了解那些关键概念。另外，轻松的行文风格也使得阅读本书是一种享受。”

——Jeff Cohen，Purple Workshops创始人

“作为Web开发人员，我自以为了解HTML和CSS。这本书让我认识到，只了解那些基础知识是不够的，Web设计决不只是改改字体和颜色那么简单。”

——Mike Weber，Web应用开发人员

“在这本以开发人员为目标读者的书中，模糊了一些公司中存在的设计团队和开发团队之间的界限。毕竟，‘程序员’也可以创造出美观易用的页面。”

——Jon Kinney，Avastone Technologies，Ruby构架师

“无论是初学者还是经验丰富的设计师，都能在这本书中有所收获。从开发人员的角度看，这本书在我的日常工作中发挥了巨大作用，它让我在组织页面内容的时候三思而后行。”

——Chris Johnson，解决方案开发专家

The Pragmatic Programmers

图灵网站：www.turingbook.com　热线：(010)51095186转604

反馈/投稿/推荐信箱：contact@turingbook.com

有奖勘误：debug@turingbook.com

分类建议 计算机/网页制作

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-25911-0

定价：59.00元